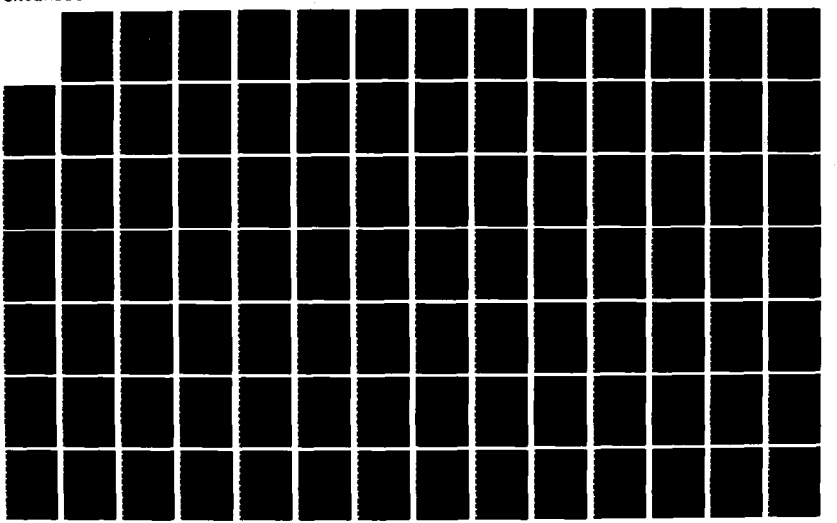
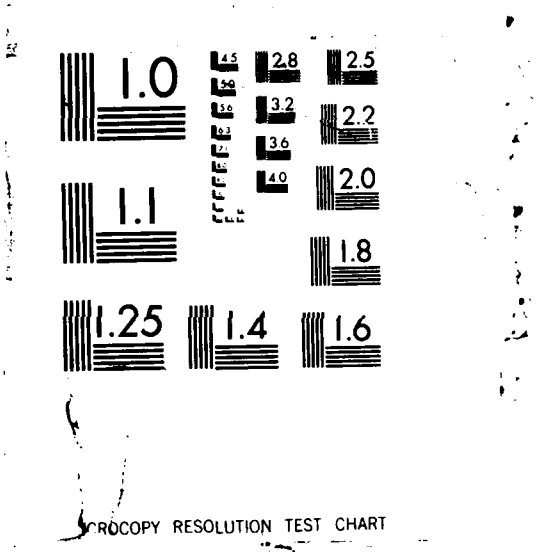
AD-A175 147 ORTHOGONAL LATTICE MODELING OF NONLINEAR SYSTEMS(U) 1/2
NAVAL POSTGRADUATE SCHOOL MONTEREY CA S L JOHNSON
SEP 86

UNCLASSIFIED F/G 9/3 NL




1.0
1.1
1.25
1.4
1.6
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.8
CROCOPY RESOLUTION TEST CHART

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

ORTHOGONAL LATTICE MODELING OF
NONLINEAR SYSTEMS

by

Scot Lee Johnson

September 1986

Thesis Advisor: S.R. Parker

Approved for public release; distribution is unlimited.

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL (*If applicable*) 62	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c ADDRESS (*City, State, and ZIP Code*) Monterey, CA 93943-5000	7b. ADDRESS (*City, State, and ZIP Code*) Monterey, CA 93943-5000
8a NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (*If applicable*)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (*City, State, and ZIP Code*)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11 TITLE (*Include Security Classification*)
ORTHOGONAL LATTICE MODELING OF NONLINEAR SYSTEMS

12 PERSONAL AUTHOR(S)
Johnson, Scott L.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (*Year, Month, Day*)	15 PAGE COUNT
Master's Thesis	FROM _____ TO _____	1986 September	143

16 SUPPLEMENTARY NOTATION

17 COSATI CODES		
FIELD	GROUP	SUB-GROUP

18 SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*)
Autoregressive; Deconvolution; Lattice Filter; Nonlinear Lattice Filter; Kalman Filter; Least-squares Inverse Filter; Nonlinear Deconvolution.

19 ABSTRACT (*Continue on reverse if necessary and identify by block number*)

The application of analysis lattice filters to the problem of determining the input to a system from observations of the system's output (i.e., deconvilution is discussed. Both linear and nonlinear systems are considered. Lattice filter modeling algorithms (Levinson and Schur) are presented.

The theory of least-squares inverse filters is reviewed. This leads to a discussion of the lattice filter, which in turn leads to the Generalized Lattice Theory. The Generalized Lattice Theory is then used to develop a nonlinear lattice structure. Simulations show that the nonlinear lattice is an effective inverse filter for both linear and nonlinear systems.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL Prof S. Parker	22b TELEPHONE (*Include Area Code*) (408)646-2788 / 22c OFFICE SYMBOL 62Px

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

Approved for public release; distribution is unlimited.

Orthogonal Lattice Modeling of Nonlinear Systems

by

Scot Lee Johnson
Lieutenant, United States Navy
B.S., University of Minnesota, 1979

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 1986

Author: Scot Lee Johnson
Scot Lee Johnson

Approved by: Sydney R. Parker
Sydney R. Parker, Thesis Advisor

Lawrence J. Ziomek
Lawrence J. Ziomek, Second Reader

H. B. Rigas, Chairman, Department of Electrical and Computer Engineering

John N. Dyer, Dean of Science and Engineering

ABSTRACT

The application of analysis lattice filters to the problem of determining the input to a system from observations of the system's output (i.e., deconvolution) is discussed. Both linear and nonlinear systems are considered. Lattice filter modeling algorithms (Levinson and Schur) are presented.

The theory of least-squares inverse filters is reviewed. This leads to a discussion of the lattice filter, which in turn leads to the Generalized Lattice Theory. The Generalized Lattice Theory is then used to develop a nonlinear lattice structure. Simulations show that the nonlinear lattice is an effective inverse filter for both linear and nonlinear systems.

TABLE OF CONTENTS

ACKNOWLEDGEMENT

I wish to thank Professor S. R. Parker for his invaluable support in the writing of this thesis. His insight, guidance, and encouragement made this work possible.

My thanks also go to Professor L. J. Ziomek for his careful review and editing of this thesis. His efforts resulted in an improved final product.

Finally, I wish to thank Major Ed Siomacco for sharing his "lessons learned" in thesis preparation. His helpful hints and suggestions were greatly appreciated.

I. INTRODUCTION

The problem of estimating a signal based upon observations of a related signal is one of the most important operations in signal processing. The input and output signals of a linear system are related by the convolution operation

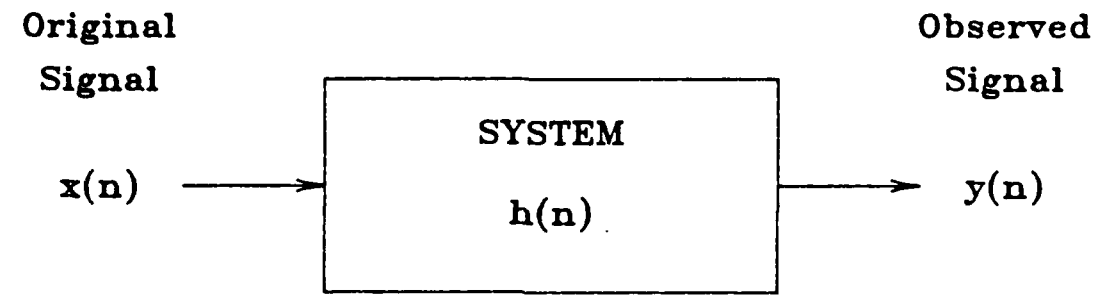
$$y(t) = h(t) * x(t) = \int_{-\infty}^{t} h(t-\tau)x(\tau)d\tau \qquad (1.1)$$

where h(t) is a causal, linear time-invariant (LTI), system impulse response. Since this thesis deals primarily with discrete digital signals, continuous time signals are sampled at uniform intervals, T, and are represented by discrete sequences x(nT) = x(n) for n = 0,1,2,...,N. Discrete convolution for a LTI system is defined by

$$y(n) = h(n) * x(n) = \sum_{m=-\infty}^{n} h(n-m)x(m) \qquad (1.2)$$

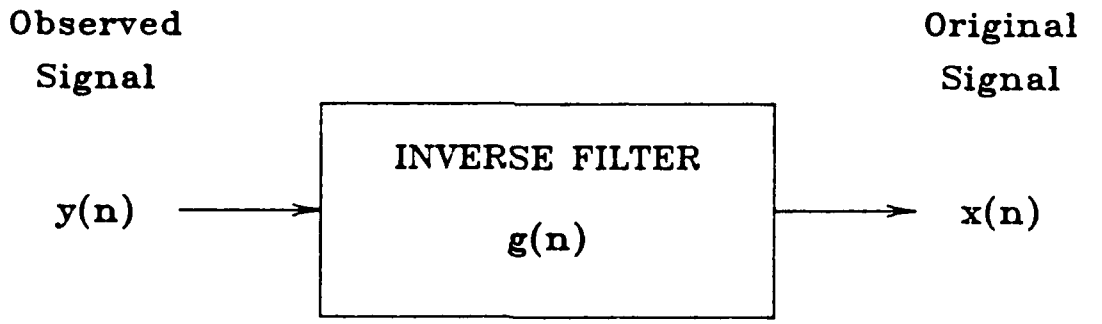
and is shown in Figure 1.1. The basic deconvolution problem is to estimate the signal x(n), assuming that both y(n) and h(n) are known. Figure 1.2 depicts the inverse filtering process of recovering the input signal from the output signal by removing the system's impulse response.

Deconvolution has important applications in a variety of fields: Radar, communications, image processing, speech



$$y(n) = x(n) * h(n)$$

Figure 1.1: Convolution



$$x(n) = y(n) * g(n)$$

Figure 1.2 Deconvolution

synthesis, seismology. For example, in image processing, deconvolution is used to recover the representation of the original object, x(m,n), from the representation of its image, y(m,n), by removing the blurring caused by the optical system's point-spread function, h(m,n). A common problem which arises in geophysics involves the deconvolution of a seismic trace, y(n), into the approximately known impulsive waveform, h(n), and the desired reflection response, x(n), which reveals the structure of the layered Earth. In other applications, h(n) may represent the impulse response of a transmission channel, magnetic recording medium, or measurement device which broadens and smears (intersymbol interference) the desired message x(n).

There have been numerous approaches to the linear deconvolution problem, including least-squares filtering, linear inverse theory, linear programming, and homomorphic signal processing. The first part of this thesis deals with least-squares filtering techniques; the application of Kalman, waveshaping, and lattice filters to deconvolution is reviewed. Next, the lattice filter discussion is extended to the theory of the generalized lattice filter. Finally, nonlinear system theory is briefly reviewed, and the nonlinear lattice filter is developed and applied to the inverse filtering problem. Computer simulation results for both the linear and nonlinear lattice filters are presented.

II. LINEAR LEAST-SQUARES DECONVOLUTION TECHNIQUES

A. INTRODUCTION

This chapter begins with a discussion of the principles of least-squares filtering theory since deconvolution is essentially an inverse filtering process. Three specific types of least-squares filters are then introduced: Kalman, waveshaping, and lattice filters. The application of each of these three filter types to the problem of linear deconvolution is studied. Finally, the chapter concludes with the presentation of computer simulation results obtained for deconvolution experiments using the lattice filter.

The goal of deconvolution is to recover the input signal, $x(n)$, to a system based upon observed values of the system's output signal $y(n)$. An optimal processor must be determined to produce the best possible estimate of $x(n)$ based upon present and past values of the output $y(n)$. A traditional measure for defining the "best" signal processor is the minimum mean-squared error criterion. Using this criterion, the estimate $\hat{x}(n)$ is defined by a linear combination of the observed values $y(n)$. Assuming causality, and that the observed signal is windowed to include only the M past samples, $\hat{x}(n)$ is written as

$$\hat{x}(n) = \sum_{i=n-M}^{n} h(n,i)y(i) \quad , \quad n \geq 0 \qquad (2.1)$$

where $h(n,i) = 0$ for $n < i$. The estimation error is given by $e(n) = x(n) - \hat{x}(n)$. To find the optimum signal processor, the $h(n,i)$ coefficients which minimize the mean-square estimation error J, where

$$J = E[e^2(n)] = E[(x(n) - \hat{x}(n))^2] \quad , \qquad (2.2)$$

must be determined. This is known as the Wiener filter formulation of the problem. [Ref. 1:pp. 113,116]

The least-squares theory of filtering began in the 1940's with the work of Norbert WIENER [Ref. 2:pp. 147-148]. WIENER developed a frequency domain procedure to design optimum filters, where optimality was defined by minimizing a mean-square error performance criterion. The Wiener filter is conventionally applied to linear time-invariant systems with stationary statistics when it is desired to separate one signal from another. In the early 1950's, the Wiener filter was extended to include time varying and nonstationary statistics, but the calculations are cumbersome [Ref. 3:p. 1].

The mean-square estimation error J(n) is minimized by setting its partial derivatives, with respect to each of the filter coefficients h(n,i), equal to zero. Thus:

$$\frac{\partial J(n)}{\partial h(n,i)} = 0 \qquad (2.3)$$

for $i = n-M, n-M+1, \ldots, n$. This yields a set of M+1 linear simultaneous equations, called the orthogonality equations where

$$R_{ey}(n,i) = E[e(n)y(i)] = 0 \qquad (2.4)$$

for $n-M \leq i \leq n$ and $n \geq 0$. $R_{ey}(n,i)$ is the crosscorrelation function between the error signal, e(n), and the data y(n). Substituting $e(n) = x(n) - \hat{x}(n)$ into the orthogonality equations gives the normal , or Wiener-Hopf equations:

$$E[x(n)y(i)] = \sum_{k=n-M}^{n} h(n,k)E[y(k)y(i)] \qquad (2.5a)$$

or

$$R_{xy}(n,i) = \sum_{k=n-M}^{n} h(n,k)R_{yy}(k,i) \quad . \qquad (2.5b)$$

Since the autocorrelation function R_{yy} of the input signal and the crosscorrelation function R_{xy} of the desired output signal with the input signal are known quantities, the M+1 equations can be solved for the optimal filter weights h(n,i), i=n-M,...,n. If data vectors are defined so that

$$\underline{x}(n) = [x(n-M), x(n-M+1), \ldots, x(n)]^T \qquad (2.6a)$$

and

$$\underline{y}(n) = [y(n-M),\ y(n-M+1),\ldots,y(n)]^T \tag{2.6b}$$

then the M+1 Wiener-Hopf equations can be written as

$$E[x(n)\underline{y}^T(n)] = \underline{h}E[\underline{y}(n)\underline{y}^T(n)] \tag{2.7}$$

where $\underline{h} = [h(n,n-M),\ h(n-M+1),\ldots,h(n,n)]$. Assuming that the signals x(n) and y(n) are stationary, then the filter coefficients are time-invariant and h(n,k) = h(n-k); and equation (2.7) can be written in matrix form as

$$\begin{vmatrix} Ryy(0) & Ryy(1) & Ryy(2) & \ldots & Ryy(M) \\ Ryy(1) & Ryy(0) & Ryy(1) & \ldots & \\ Ryy(2) & Ryy(1) & Ryy(0) & \ldots & \\ \vdots & \vdots & \vdots & \ddots & \\ Ryy(M) & & & & Ryy(0) \end{vmatrix} \begin{vmatrix} h(0) \\ h(1) \\ h(2) \\ \vdots \\ h(M) \end{vmatrix} - \begin{vmatrix} Rxy(0) \\ Rxy(1) \\ Rxy(2) \\ \vdots \\ Rxy(M) \end{vmatrix} = \tag{2.8}$$

Now, the optimal coefficients can be solved by inverting the autocorrelation matrix, or by exploiting the matrix's Toeplitz structure (all elements are the same on any given diagonal) to employ the more efficient Levinson algorithm. (Levinson's algorithm will be discussed in the development of the analysis lattice filter.)

The minimized value of the mean-square estimation error can now be computed, and is found to be

$$J_{min}(n) = E[x^2(n)] - \sum_{i=n-M}^{n} h(n,i)E[y(i)x(n)] \tag{2.9}$$

$$= R_{xx}(0) - E[x(n)\underline{y}^T(n)]E[\underline{y}(n)\underline{y}^T(n)]^{-1}E[\underline{y}(n)x(n)]$$

[Ref. 4:p. 148]. These values correspond to the diagonal elements of the covariance matrix of the estimation error,

$$R_{ee} = E[\underline{e}(n)\underline{e}^T(n)] \tag{2.10}$$

where $\underline{e}(n) = \underline{x}(n) - \hat{\underline{x}}(n)$ [Ref. 1:p. 120]. Furthermore, the estimate of x(n) is given by

$$\hat{x}(n) = E[x(n)\underline{y}^T(n)]E[\underline{y}(n)\underline{y}^T(n)]^{-1}\underline{y}(n) \tag{2.11}$$

$$= \underline{h}\underline{y}(n)$$

This estimate can be thought of as the projection of the desired signal x(n) onto the space spanned by the components of the observation vector $\underline{y}(n)$. The minimized estimation error vector is orthogonal, or normal, to the estimate $\hat{x}(n)$. [Ref. 4:p. 147]

This completes the overview of least-squares filtering. The following sections present several linear deconvolution techniques which employ this criterion: Kalman filtering, spiking filters, and lattice filters.

B. KALMAN FILTER

In the early 1960's, R.E. KALMAN introduced an optimal recursive filter based on state-space time-domain methods [Ref. 2:pp. 267-268]. The Kalman filter estimates the state

of a linear system, and is optimal in the sense that it minimizes the mean-square error of the state estimate. The Kalman filter is useful when the system is defined by state space equations: The system signals are represented by random processes, and the data observations are contaminated by noise. The Kalman filter algorithm processes measurement data, and requires a priori state space models (known or assumed) of the system and measurement dynamics. Also, the statistics of the system input and measurement noises, as well as initial condition information, are required to produce the state estimate. This process is depicted in Figure 2.1. Here, the discrete Kalman filter equations will be presented, and then their application to deconvolution will be described.

The state space representation of the discrete system and measurement models (see Figure 2.2) are written as [Ref. 2:pp. 195-200]:

$$\underline{x}(k) = F(k-1)\underline{x}(k-1) + \underline{w}(k-1) \tag{2.12}$$

$$\underline{z}(k) = H(k)\underline{x}(k) + \underline{v}(k) \tag{2.13}$$

where

$\underline{x}(k)$ = (n x 1) system state vector

$F(k)$ = (n x n) transition matrix

$\underline{w}(k)$ = (n x 1) system noise vector

$\underline{z}(k)$ = (m x 1) measurement vector

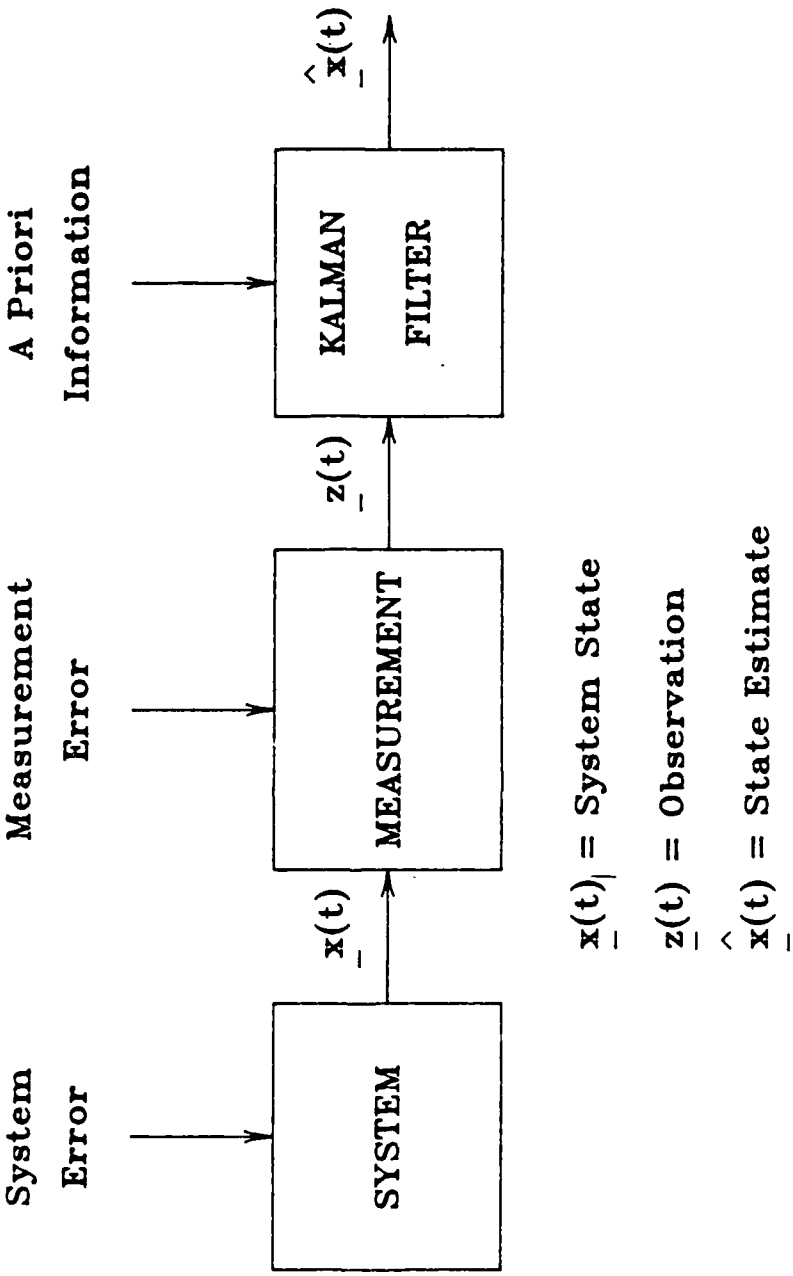


Figure 2.1 System, Measurement, and Kalman Filter

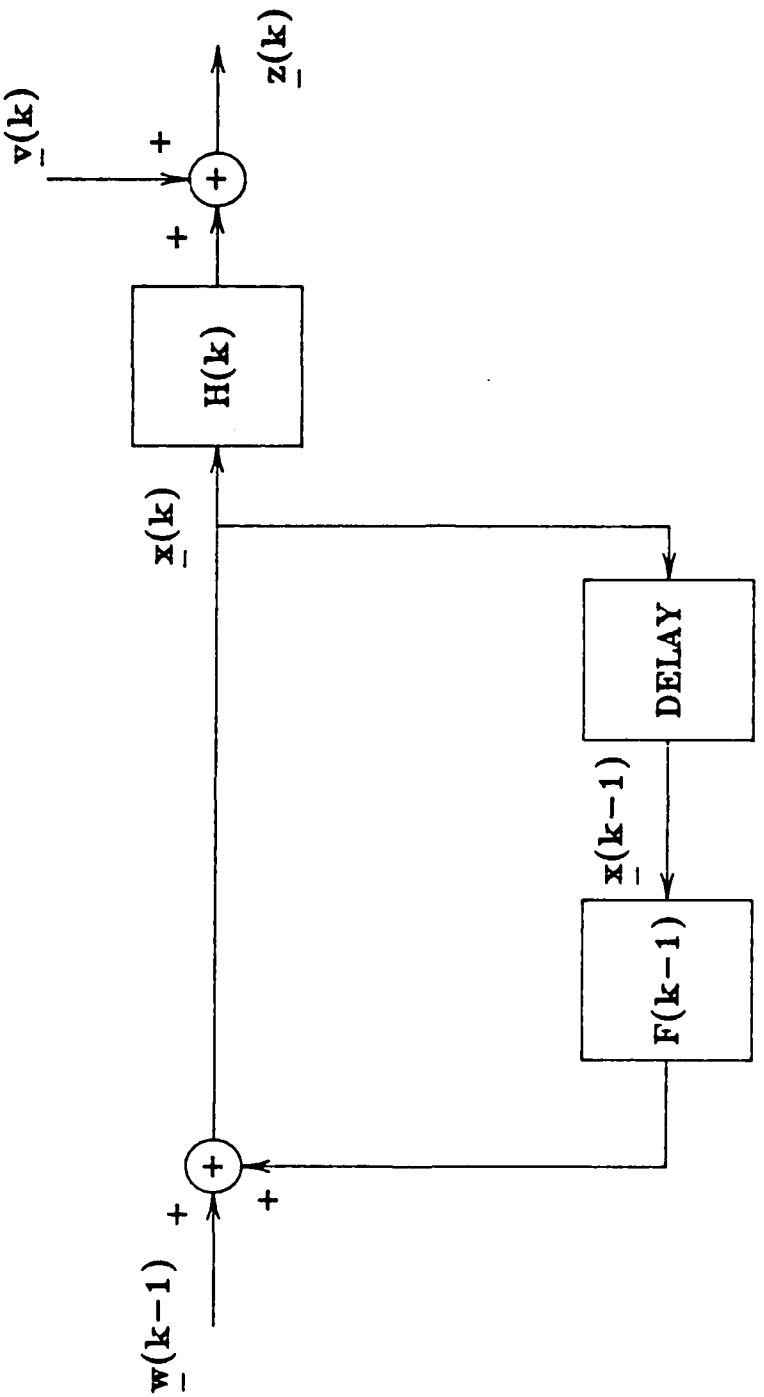


Figure 2.2 System and Measurement Model

$H(k)$ = (m x n) observation matrix

$\underline{v}(k)$ = (m x 1) measurement noise vector.

(Note that the time index k is used here to be consistent with the discrete Kalman filter literature.)

The noise vectors $\underline{w}(k)$ and $\underline{v}(k)$ are assumed to have zero mean, white, Gaussian distributions with covariance matrices of $Q(k) = E[\underline{w}(k)\underline{w}^T(k)]$ and $R(k) = E[\underline{v}(k)\underline{v}^T(k)]$, respectively. Additionally, $\underline{w}$ and $\underline{v}$ are uncorrelated so that $E[\underline{w}(k)\underline{v}^T(j)] = 0$ for all k and j. The state estimation error is defined by

$$\underline{e}(k|k-1) = \underline{x}(k) - \hat{\underline{x}}(k|k-1) \tag{2.14}$$

and the associated (n x n) error covariance matrix is

$$P(k|k-1) = E[\underline{e}(k|k-1)\underline{e}^T(k|k-1)]. \tag{2.15}$$

An updated, a posteriori estimate of $\underline{x}(k)$ is obtained from the measurement $\underline{z}(k)$ and the a priori state estimate $\hat{\underline{x}}(k|k-1)$ by

$$\hat{\underline{x}}(k) = \hat{\underline{x}}(k|k-1) + K(k)[\underline{z}(k)-H(k)\hat{\underline{x}}(k|k-1)] \tag{2.16}$$

where $K(k)$ is the (n x m) Kalman gain matrix. The a posteriori error is given by

$$\underline{e}(k) = \underline{x}(k) - \hat{\underline{x}}(k) \tag{2.17}$$

and the associated a posteriori error covariance matrix is

$$P(k) = E[\underline{e}(k)\underline{e}^{T}(k)]. \tag{2.18}$$

The mean-square estimation error criterion is minimized when

$$K(k)=P(k|k-1)H^{T}(k)[H(k)P(k|k-1)H^{T}(k)+ R(k)]^{-1} \tag{2.19}$$

This optimal value of the Kalman gain matrix minimizes the individual terms along the main diagonal of P(k). Substituting the optimal gain matrix K(k) into the expression for P(k) results in

$$P(k) = (\underline{I} - K(k)H(k))\ P(k|k-1)\ , \tag{2.20}$$

where $\underline{I}$ is the identity matrix. In order to compute equations (2.16) and (2.19) recursively, the a priori estimates $\hat{\underline{x}}(k+1|k)$ and P(k+1|k) must be determined at time k. The a priori estimates are given by

$$\hat{\underline{x}}(k+1|k) = F(k)\hat{\underline{x}}(k) \tag{2.21}$$

$$\begin{aligned} P(k+1|k) &= E[\underline{e}(k+1|k)\underline{e}^{T}(k+1|k)] \\ &= E[F(k)\underline{e}(k)+\underline{w}(k))(F(k)\underline{e}(k)+\underline{w}(k))^{T}] \\ &= F(k)P(k)F^{T}(k) + Q(k)\ . \end{aligned} \tag{2.22}$$

The Kalman filter algorithm is implemented by recursively computing equations (2.16), (2.19), (2.20), (2.21), and (2.22). Figure 2.3 is a diagram of the Kalman filter. The

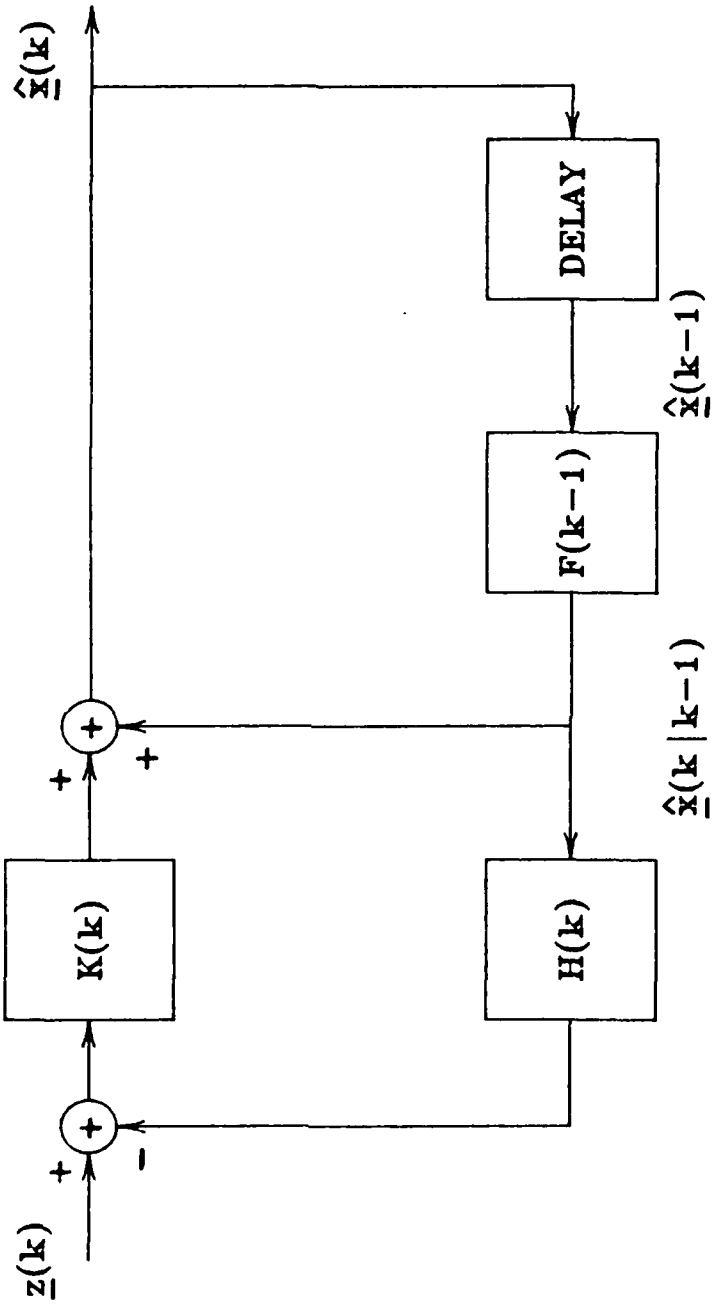


Figure 2.3 Discrete Kalman Filter

algorithm is initiated with the initial conditions

$$\hat{\underline{x}}(0) = E[\underline{x}(0)] \tag{2.23a}$$

and,

$$P(0) = E[(\underline{x}(0)-\hat{\underline{x}}(0))(\underline{x}(0)-\hat{\underline{x}}(0))^{T}]. \tag{2.23b}$$

In the event that a controlling input or a deterministic disturbance $\underline{u}(k)$ is applied to the system, the only change in the above algorithm is the state model and the a posteriori estimate. They become [Ref. 5:p. 130]

$$\underline{x}(k) = F(k-1)\underline{x}(k-1) + \underline{w}(k-1) + G(k-1)\underline{u}(k-1) \tag{2.24}$$

$$\hat{\underline{x}}(k) = F(k-1)\hat{\underline{x}}(k-1)+G(k-1)\underline{u}(k-1) + K(k)[\underline{z}(k)-H(k)F(k-1)\hat{\underline{x}}(k-1)]. \tag{2.25}$$

where $G(k)$ is a $(n \times q)$ matrix and $\underline{u}(k)$ is a $(q \times 1)$ vector of input signals.

The problem of estimating a desired signal based on noisy data observations pertains to such fields as communications, controls, and geophysics. However, the Kalman filter can also be applied to these deconvolution problems. As an example, the discrete Kalman inverse filter will now be applied to an exploration seismology problem.

Typically, in the search for underground oil and gas deposits , a vibratory signal source generates a pulse of energy which is transmitted into the earth. In modern

seismology, the shape of this source wavelet can be carefully controlled; it is chosen so that it contains only those frequencies which are transmitted best by the earth. As the source wavelet propagates through the earth, it encounters many different layers with various acoustic impedances. At these layers, both partial reflection and refraction occur, creating numerous transmission paths. The received seismic signal at the surface is composed of many overlapping reflected wavelets. Therefore, the seismic trace can be represented as the convolution of the original source wavelet with an impulse train representing the various layers of the earth. Moreover, the seismic trace is contaminated by measurement noise and by the phenomena of ghost reflections and reverberations. [Ref. 6:pp. 14-15]

The seismic trace can be described mathematically by

$$z(t) = s(t,T)*r(t) + v(t) = \int_{t_o}^{t} s(t,T)r(T)dT + v(t) \quad (2.26)$$

where $z(t)$ = measured seismic trace

$s(t,T)$ = finite duration, time varying wavelet

$r(t)$ = reflectivity function of earth's structure

$v(t)$ = measurement noise.

The seismologist must extract the structure of the earth, $r(t)$, by analyzing the noisy seismic data $z(t)$. This process of removing the wavelet shape from the trace and leaving behind the impulse train representing the reflected

wavelet's strength and arrival time is that of deconvolution. CRUMP [Ref. 3:pp. 6-7] shows that if the seismic signal is sampled at discrete, uniformly spaced intervals, and if s(t,T) and r(t) are assumed to be causal, then z(t) is represented by

$$z(k) = \sum_{i=1}^{J} [s(k,k-i+1)r(k-i+1)] + v(k) \qquad (2.27)$$

where the sample number k = 1,2,3,..., and

L = length of the wavelet given in number of samples

J = k for k<L

= L for k≥L .

When M traces of K samples in length are available for processing then $\underline{z}(k)$ becomes a (M x 1) vector where the j-th component is given by

$$z_j(k) = \sum_{i=1}^{J} [H_{ji}(k))r(k-i+1)] + v_j(k) \qquad (2.28)$$

for j = 1,2,...,M and k = 1,2,...,K. This assumes that the reflectivity function is the same for each trace, while the shape of the exciting wavelet may vary from trace to trace. The time-varying wavelet values are contained in the (M x L) matrix H(k) where the j-th row contains the L samples of the wavelet which generates the j-th trace:

$$H_{ji}(k) = s_j(k,k-i+1) \; . \qquad (2.29)$$

In vector form, the equations become

$$\underline{z}(k) = H(k)\underline{x}(k) + \underline{v}(k) \tag{2.30}$$

where the state, measurement, and noise vectors are given by

$$\underline{x}(k) = [r(k), r(k-1), \ldots, r(k-L+1)]^T \tag{2.31a}$$

$$\underline{z}(k) = [z_1(k), z_2(k), \ldots, z_M(k)]^T \tag{2.31b}$$

$$\underline{v}(k) = [v_1(k), v_2(k), \ldots, v_M(k)]^T . \tag{2.31c}$$

This is the Kalman filter measurement model. Now the state model must be determined.

The state model is arrived at by assuming a general relationship for the reflection coefficients of $\underline{x}(k)$. The assumed relationship is the autoregressive equation

$$r(k) = \sum_{i=1}^{L} [b_i(k-1)r(k-i)] + w(k-1) . \tag{2.32}$$

Comparing equation (2.32) with the state vector $\underline{x}(k)$ yields the state model

$$\underline{x}(k) = F(k,k-1)\underline{x}(k-1) + \underline{w}(k-1) \tag{2.33}$$

where the (M x L) transition matrix and the (M x 1) system noise vector are given by

$$F(k,k-1) = \left[\begin{array}{ccc:c} b_1(k-1) & b_2(k-1) & \ldots & b_L(k-1) \\ \hline & & & \\ & \underline{I} & & \underline{0} \\ & & & \end{array}\right] , \tag{2.34a}$$

$$\underline{w}(k-1) = [w(k-1),0,0,\ldots,0]^T , \qquad (2.34b)$$

$\underline{I}$ is the identity matrix, and $\underline{0}$ is the null vector. Equations (2.30) and (2.32) provide the measurement and system models for implementing the deconvolution via the Kalman filter. CRUMP discusses methods by which to obtain numerical values for the reflection coefficient vector $\underline{b}(k)$ and the time-varying wavelet sample matrix H(k) [Ref. 3:pp 8-11]. Once these matrices are determined, the recursive Kalman filter removes the effects of s(t,T) from r(t) and generates the state estimate $\underline{x}(k)$ which provides L samples of the desired reflectivity function at each time k.

It is interesting to note that both the Kalman and Wiener filters are minimum mean-square error estimators, both require the same a priori knowledge of the process to be estimated, and that both yield identical estimates. However, the Kalman filter does have distinct advantages over the Wiener filter. First, due to the matrix form of the state space equations, the Kalman filter has multichannel capability and is equivalent to a bank of optimal estimators. Moreover, the Kalman filter is ideally suited to computer implementation due to its discrete and recursive characteristics. [Ref.2:pp. 268-269]

The discrete least-squares approach which follows is a viable alternative to the Kalman filter algorithm,

particularly when state space model equations are not available or applicable.

C. LEAST-SQUARES INVERSE FILTER

When a linear system, H(z), is excited by the an input signal x(n), the output y(n) is defined by the convolution relationship $y(n) = x(n) * h(n)$. Deconvolution involves finding the inverse filter G(z) such that $H(z)G(z) = 1$. (Note that the discrete time domain is related to the frequency domain through the equation $z = e^{jwT}$, where T is the discrete sampling interval.) This condition transforms to $h(n) * g(n) = d(n)$ in the discrete time domain; h(n) and g(n) are the impulse responses of the filters H(z) and G(z), respectively, and d(n) is the unit impulse function. If this condition is met, then the original input signal x(n) is recovered at the output of the inverse filter.

The transfer function of the causal system is defined by the infinite series obtained by taking the one-sided z-transform of the system's impulse response:

$$H(z) = Y(z)/X(z) = \sum_{i=0}^{\infty} h(i)z^{-i} . \qquad (2.35)$$

H(z) can be determined by inserting a known sequence x(n) into the system, measuring the output sequence y(n), and then manipulating their z-transforms. The inverse filter G(z) is then computed by carrying out the polynomial division

$$G(z) = 1/H(z) = 1/[h(0)+h(1)z^{-1}+h(2)z^{-2}+\ldots] \qquad (2.36)$$

$$= g(0) + g(1)z^{-1} + g(2)z^{-2} + \ldots + g(M)z^{-M} + \ldots$$

and truncating to M+1 terms if necessary. If the exact inverse filter is approximated by truncating G(z) to order M, then its impulse reponse is given by the sequence g(n), n=0,1,2,...,M. Furthermore, if the original system's impulse response is represented by h(n), n=0,1,2,...N, then

$$d(n) \cong g(n) * h(n) = \sum_{m=0}^{M} g(m)h(n-M) \qquad (2.37)$$

for $0 \leq n \leq N+M$. This approximation to the impulse function improves as the order M of the inverse filter is increased.

Now the stability of the inverse filter will be addressed. If H(z) has all its zeros inside the unit circle in the complex z-plane, it is referred to as a minimum-delay polynomial; the corresponding sequence h(n) is called a minimum phase-lag sequence. This is a sufficient condition to guarantee that H(z) has a stable inverse, because the zeros of H(z) become the poles of G(z), and G(z) is a stable filter if all its poles lie within the unit circle. Maximum- and mixed-delay signals are obtained by transforming the zeros of H(z) from z_i to $1/z_i^*$ where the superscript '*' represents the complex conjugate operation. A maximum-delay sequence has all of its zeros outside the

unit circle, while the mixed-delay sequence has zeros inside and outside the unit circle.

If H(z) has N zeros, then transforming these zeros results in at most 2^N distinct sequences. Of note, each of these minimum, maximum, and mixed-delay signals have the same magnitude spectrum: $|H(w)|^2 = H(w)H^*(w)$ [Ref. 1:p. 98]. However, they do have distinct phase spectra [Ref. 4:p. 175]. The maximum-delay polynomial can be written as

$$H^R(z) = h^*(N) + h^*(N-1)z^{-1} + \ldots + h^*(0)z^{-N} \qquad (2.38)$$

The so called reverse polynomial $H^R(z)$ is a conjugated, reflected, and shifted version of H(z). The corresponding maximum phase-lag sequence is $\underline{h}^R = \{h^*(N), h^*(N-1), \ldots, h^*(0)\}$ [Ref. 6:p. 72] While the minimum-delay filter has a causal, stable inverse consisting only of a memory function, the maximum-delay filter has an inverse which consists only of a stable, noncausal, anticipation function. The stable inverse of a mixed-delay function consists of both memory and anticipation functions. Filters with nonvanishing anticipation components are noncausal; they cannot work in real time since the future values of the filter input are not available for processing. This problem can be circumvented if the entire signal is first recorded prior to analysis; then the required future input data is available. [Ref. 4:p. 87]

The energy distribution in minimum, mixed, and maximum delay signals will now be examined. Since each of these signals have an identical magnitude spectrum, they also have the same total energy. However, although the total energy is the same, the rate at which the energy builds up differs for the various sequences. Parseval's theorem states that the total energy in a signal is given by

$$\int_{-\pi}^{\pi} |H(w)|^2 \, dw/(2\pi) = \sum_{m=0}^{M} |h(m)|^2 . \qquad (2.39)$$

If the partial energy is defined as

$$P(n) = \sum_{m=0}^{n} |h(m)|^2 , \qquad (2.40)$$

then it can be shown that the energy builds up quickest in the minimum-delay sequence, and that it builds up the slowest in the maximum-delay sequence [Ref. 6:pp. 75-76]. In other words, the minimum-delay signal makes its impact as soon as possible since its energy is concentrated at the front of the sequence. The maximum-delay signal makes its major impact at a later time since its energy is concentrated at the end of the sequence. The energy curves associated with all the possible mixed-delay signals lie between these two extremes. Finally, it can be shown that the convolution of two minimum-delay sequences with one another results in a minimum-delay sequence. The convolution of maximum-delay signals results in a maximum-delay

sequence. Convolution involving any other combination of sequences results in a mixed-delay sequence. Of note, the resulting maximum and minimum-delay sequences are the reverse of each other. [Ref. 6:pp. 73-74]

The method described above of finding an approximate, finite length inverse filter consisted of simply truncating the exact inverse filter found by polynomial division. It was seen that the inverse filter G(z) attempted to transform the impulse response of H(z) into a unit impulse located at the origin. This can be thought of as an attempt by G(z) to undo the blurring effect of H(z) (i.e., H(z) "blurs" the impulse d(n) into the impulse response h(n)). If the input to H(z) is designated x(n), and if the output of G(z) is $\hat{x}(n)$, then the error of the approximated inverse filter is

$$e(n) = x(n) - \hat{x}(n) \text{ for } 0 \leq n \leq N+M. \tag{2.41}$$

The error energy is defined by

$$J = \sum_{n=0}^{N+M} e^2(n) \tag{2.42}$$

For the polynomial division / truncation method, J decreases as the order M of G(z) increases [Ref. 6:p. 136]. Seeking to minimize the error energy J with respect to the inverse filter coefficients leads to another approach for finding an approximate inverse filter.

Filters which minimize the mean-square error J are called least error energy or least-squares inverses. In general, the desired output sequence x(n) of the inverse filter G(z) could be of any shape. G(z) is then called a waveshaping filter. When applied to deconvolution, the desired output of the inverse filter is a unit impulse d(n-i), where i = 0,1,2,...,N+M defines the lag of the digital inverse filter. In this case, G(z) is called an i-th delay spiking filter since it tries to condense the system impulse response h(n) into a spike with i delays. Since i = 0,1,2,...,N+M, there are N+M+1 possible spiking filters. As will be discussed, there are preferred values of the lag i for minimizing J, depending on the phase characteristics of h(n). The spiking filter problem is shown in Figure 2.4.

It is convenient to restate the deconvolution problem in a matrix form based on the Yule-Walker, or autocorrelation, method [Ref. 1:pp. 243-245]. This is accomplished by defining the (M+N+1 x M+1) system output matrix Y, the (M+1 x 1) impulse response vector $\underline{g}$, and the (N+M+1 x 1) input vector $\underline{x}$ as follows:

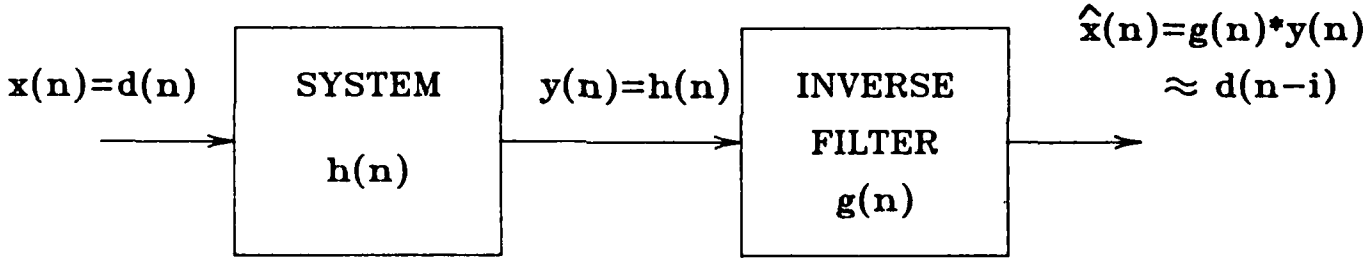


Figure 2.4 Spiking Filter

d(n) = unit impulse function
h(n) = impulse response of system
g(n) = impulse response of inverse filter

$$
Y = \begin{vmatrix}
 & \overleftarrow{\text{M ZEROS}} & & & \overrightarrow{\phantom{\text{M ZEROS}}} \\
y(0) & 0 & 0 & \cdots & 0 \\
y(1) & y(0) & 0 & \cdots & 0 \\
y(2) & y(1) & y(0) & \cdots & 0 \\
\vdots & \vdots & \vdots & & \\
\vdots & \vdots & \vdots & \cdots & 0 \\
y(N) & y(N-1) & y(N-2) & \cdots & y(N-M) \\
0 & y(N) & y(N-1) & & \\
0 & 0 & y(N) & & \\
\vdots & \vdots & 0 & \ddots & \\
\vdots & \vdots & \vdots & \ddots & \\
0 & 0 & 0 & & y(N)
\end{vmatrix} \qquad (2.43a)
$$

$$
\underline{g} = [g(0),\ g(1),\ \ldots\ ,g(M)]^T\ , \qquad (2.43b)
$$

$$
\underline{x} = [x(0),\ x(1),\ \ldots\ ,x(N+M)]^T\ . \qquad (2.43c)
$$

The above notation is for the general case of the waveshaping filter. For the special case of the spiking filter, the y(n) components of the system output matrix Y are replaced by the impulse response h(n). Also, the desired signal $\underline{x}$ reduces to an impulse d(n-i). Now equations (2.37), (2.41), and (2.42) can be rewritten as

$$
\hat{\underline{x}} = Y\underline{g}\ , \qquad (2.44a)
$$

$$
\underline{e} = \underline{x} - \hat{\underline{x}}\ , \text{ and} \qquad (2.44b)
$$

$$
J = \underline{e}^T\underline{e}\ . \qquad (2.44c)
$$

As discussed in the introduction to this chapter, the least-squares criterion is satisfied by minimizing J with respect to the inverse filter coefficients G. This results in the normal equations

$$Y^T Y\underline{g} = Y^T \underline{x} \tag{2.45}$$

which can be solved for $\underline{g}$. By recognizing that $Y^T Y$ is equivalent to the (M+1 x M+1) sampled autocorrelation matrix $R = E[\underline{y}\underline{y}^T]$ where the M+1 length data vector is $\underline{y} = [\ y(0), y(1),\ldots,y(N),0,0,\ldots 0]^T$, and that $Y^T \underline{x}$ is a length (M+1) cross-correlation column vector $\underline{r}$, it can be seen that

$$\underline{g} = (Y^T Y)^{-1} Y^T \underline{x} = R^{-1} \underline{r} \tag{2.46}$$

where R^{-1} can be evaluated efficiently by Levinson's algorithm. The actual filter output is then

$$\hat{\underline{x}} = Y\underline{g} = (YR^{-1} Y^T)\underline{x} = P\underline{x} \tag{2.47}$$

where $P = Y(Y^T Y)^{-1} Y^T$ is a square (N+M+1) dimensioned matrix called the projection or performance matrix. The filter's performance improves as P approaches the identity matrix. Now, the estimation error and cost function are written as

$$\underline{e} = \underline{x} - \hat{\underline{x}} = \underline{x}(\underline{I} - P) \tag{2.48}$$

$$J = \underline{e}^T \underline{e} = \underline{x}^T (\underline{I} - P)\underline{x} \tag{2.49}$$

Since in the deconvolution problem $\underline{x}$ is the spike with i delays, the inverse filter output $\underline{x}$ is actually the i-th column of the P matrix. Also, since $\underline{g} = R^{-1} Y^T$, the coefficients of the i-th spiking filter are contained in the i-th column of the matrix $R^{-1} Y^T$. Moreover, the energy

error of the i-th spiking filter reduces to J = 1 - P(i,i). Therefore, in order to realize the best inverse function (i.e., minimize J), select the delay i for which P(i,i) is largest. For a chosen lag i, the filter's performance also improves as the order M of the inverse filter is increased.

Now let J(i) represent the estimation error for the i-th spiking filter. The grand sum of squared errors is defined as V = J(0) + J(1) +...+ J(M+N). It can be shown that V = (M+N+1) - (M+1) = N, where N is the order of the system H(z) [Ref. 4:p. 198]. Therefore, V is independent of order M.

For sufficiently long spiking filters, the optimal value of the delay i depends on the phase characteristics of the signal h(n) and the choice of the lag is governed by the following rules. If h(n) is a minimum-delay input, the spiking filter should have zero delay, i = 0. This says that for a signal with its energy concentrated towards the front of the sequence, it is easiest to condense it to a unit impulse at the origin. If the signal is a maximum-delay input, the maximum-delay spike i = N+M+1 should be selected. If h(n) is a mixed-delay signal, then the i corresponding to the largest P(i,i) should be chosen. [Ref 6.:p. 152]

Under certain conditions, the estimation error will go to zero as the length of the inverse filter tends to infinity. First, as previously discussed, if h(n) is a

minimum-delay sequence then an exact inverse can be found through polynomial division. The zero-delay spiking filter approaches this exact inverse filter G(z) as the number of terms M+1 increases. Therefore, the estimation error goes to zero as M goes to infinity. The second condition is, if h(n) is not a minimum-delay sequence, J will approach zero as 1/M if the lag i of the spiking filter is chosen to be sufficiently large. [Ref.4:pp. 200-201]

Up until now, the effects of measurement noise and imperfect knowledge of the distorting function H(z) have been neglected. If noise is introduced, then the output y(n) of the system H(z) driven by input signal x(n) becomes

$$y(n) = h(n)*x(n) + v(n) \tag{2.50}$$

where v(n) is assumed to be zero-mean white noise with variance Q. If this signal is passed through the previously determined inverse filter G(z), the filter output becomes

$$\hat{x}(n) = g(n)*y(n) = g(n)*h(n)*x(n) + g(n)*v(n) \tag{2.51}$$

$$\cong d(n)*x(n) + u(n) \cong x(n) + u(n)$$

where u(n) = g(n)*v(n) is the filtered noise signal and d(n) is the unit impulse function. The variance of u(n) is:

$$E[u^2(n)] = Q \sum_{n=0}^{M} g^2(n) = Q\underline{g}^T\underline{g} \tag{2.52}$$

[Ref. 1:p. 248]. This variance may be larger than the

original noise variance Q. For example, if h(n) is a low frequencey signal, then g(n) must be very spiky in order to compress h(n) into a high frequency content spike. As a result, g(n) will likely have values greater than one. Therefore, the variance of u(n) will be larger than Q. This further degrades the estimate $\hat{x}(n)$.

To compensate for this filtered noise, the minimization criterion is modified so that

$$J = \sum_{n=0}^{N+M} (d(n) - g(n)*h(n))^2 + \lambda Q \underline{g}^T \underline{g} \qquad (2.53)$$

where λ is a positive parameter. The first term of the cost function tries to produce a good inverse filter whereas the second term tries to reduce the output noise. If λ is large, noise reduction is emphasized at the expense of obtaining a good inverse function. A small λ emphasizes finding a good inverse function g(n), and there is little output noise reduction.

Using this new cost function, the resulting normal equations are given by

$$(Y^T Y + \lambda Q \underline{I})\underline{g} = Y^T \underline{x} \; . \qquad (2.54)$$

Comparing this with equation (2.45) reveals that the main diagonal elements of the sampled autocorrelation matrix R have been modified by an additive term: R(0) becomes R(0) + λQ. If the Backus-Gilbert "prewhitening" parameter epsilon

is introduced, where

$$\epsilon = \lambda Q/R(0) \ , \qquad (2.55)$$

then the diagonal elements are written as $(1 + \epsilon)R(0)$. Even very small values for the BG parameter have a beneficial effect of stabilizing the inverse of the matrix $(Y^T Y + \lambda Q\underline{I})$. [Ref. 1:p. 246]

D. LATTICE FILTER

1. Introduction

The lattice filter is another optimal least-squares predictor which can be applied to the linear deconvolution problem. The lattice, or ladder filter derives its name from the cascade form of its signal flow graph. Basically, the lattice filter provides an alternative to the transversal filter for modeling a signal. It can be viewed as a Gram-Schmidt orthogonalization of the incoming data. This will be elaborated upon in subsequent paragraphs. To date, much of the work done with lattice filters has been in the areas of speech analysis/synthesis, seismology, and in high-resolution spectral estimation [Ref. 7:p. 841]. Prior to developing the lattice filter equations, key mathematical concepts which apply to this discussion will be reviewed.

2. Mathematical Background

A central concept behind the development of the lattice filter is that of orthogonality. Two vectors are

orthogonal if their inner product is zero. The geometric interpretation of this is that the vectors are at right angles to one another. As an example, if the random variables u and v are the basis vectors of a two-dimensional space, they are orthogonal if and only if their inner product $\langle u,v \rangle = E[uv] = 0$. In the case where the linear space is spanned by the random variables of a data vector, two vectors are orthogonal if the corresponding random variables are uncorrelated and if one or both have zero mean [Ref. 8:p. 92].

A related theorem is the orthogonal decomposition theorem, which states that any random variable may be decomposed uniquely with respect to a subspace S into two mutually orthogonal parts, one part which is parallel to S (i.e., lies in S) and the other part orthogonal to S. That is, $\underline{y}$ can be written $\underline{y} = \hat{\underline{y}} + \underline{e}$ where $\underline{e}$ is orthogonal to $\hat{\underline{y}}$ and to the basis vectors which span the subspace S, and where $\hat{\underline{y}}$ is defined by a linear combination of the basis vectors. If S is spanned by the basis vectors $\{u(1),u(2),\ldots,u(M)\}$, then $\hat{\underline{y}} = \sum_{i=1}^{M} a(i)u(i)$ where it can be shown that the coefficients are given by the equation $a(i) = E[\underline{y}u(i)]E[u^2(i)]^{-1}$. [Ref. 1:p. 13]

The orthogonal projection theorem adds to this by stating: the orthogonal projection $\hat{\underline{y}}$ of a vector $\underline{y}$ onto a linear subspace S is that vector in S which lies closest to

$\underline{y}$ with respect to the distance induced by the inner product of the vector space [Ref. 1:p. 15]. Restated, this simply means that $\hat{\underline{y}}$ is the best estimate of $\underline{y}$ that can be made by a linear combination of the basis vectors of S in a minimum mean-squared error sense. Figure 2.5 shows the projection of $\underline{y}$ into the subspace S.

Another important concept in developing the lattice filter is that of Gram-Schmidt orthogonalization. The Gram-Schmidt procedure is a recursive orthogonalization process which generates a set of mutually orthogonal basis vectors $\{u(1),u(2),\ldots,u(M)\}$ from a given set of basis vectors $\{y(1),y(2),\ldots,y(M)\}$. The procedure is initialized by letting $u(1) = y(1)$. Next, $y(2)$ is decomposed with respect to $u(1)$. That part of $y(2)$ which is orthogonal to $u(1)$ becomes $u(2)$. Next, $y(3)$ is decomposed with respect to the subspace spanned by $\{u(1),u(2)\}$. That part of $y(3)$ which is orthogonal to this subspace becomes $u(3)$. In general, the new set of orthogonal basis vectors is defined by

$$u(n) = y(n) - \sum_{i=1}^{n-1} b(n,i)u(i) \qquad (2.56)$$

for $2 \leq n \leq M$ and where $b(n,i) = E[y(n)u(i)]E[u^{2}(i)]^{-1}$. Using the orthogonal decomposition theorem, this is equivalent to $y(n) = \hat{y}(n) + u(n)$ where $\hat{y}(n)$ is the best estimate of the n-th component of the data vector based on the previous n-1 components and $u(n)$ represents the

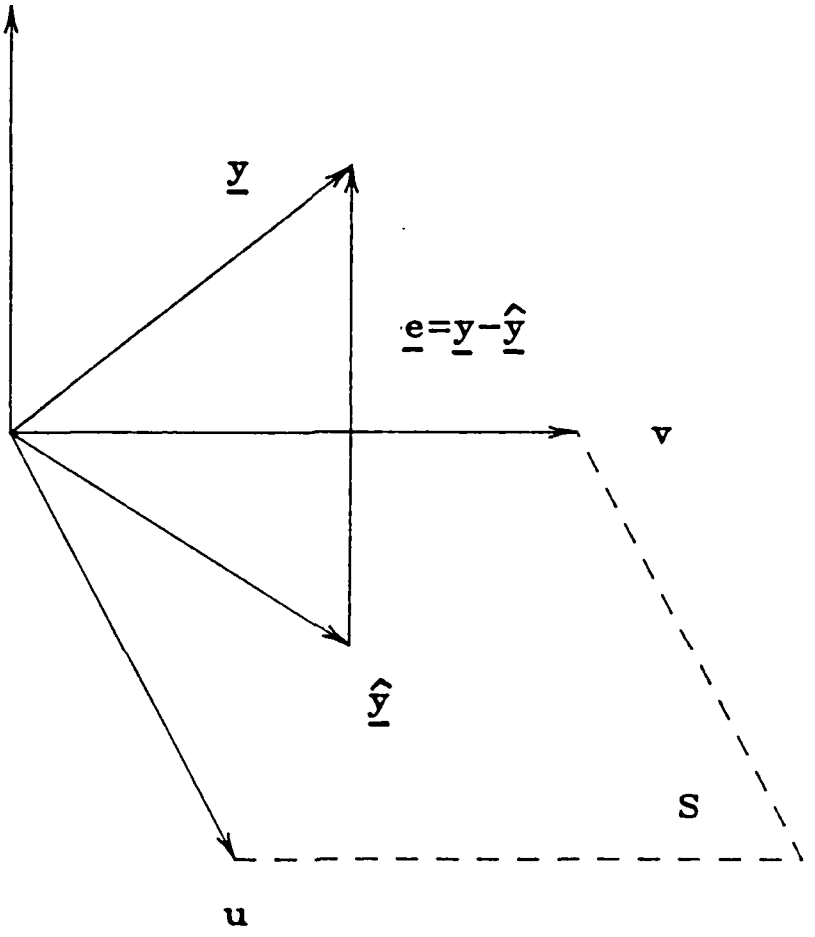


Figure 2.5 Projection of $\underline{y}$ onto Space S

estimation error. If b(n,n) = 1 is introduced, then

$$y(n) = \sum_{i=1}^{n} b(n,i)u(i) \quad \text{for } 1 \leq n \leq M. \tag{2.57a}$$

This can be written in matrix form as

$$\underline{y} = B\underline{u} \quad \text{where} \tag{2.57b}$$

$$\underline{y} = [y(1),\ y(2),\ldots,\ y(M)]^T ,$$

$$\underline{u} = [u(1),\ u(2),\ldots,\ u(M)]^T ,$$

$$B = \begin{vmatrix} 1 & 0 & 0 & \ldots & 0 \\ b(2,1) & 1 & 0 & \ldots & 0 \\ b(3,1) & b(3,2) & 1 & \ldots & 0 \\ \vdots & \vdots & & & \vdots \\ b(M,1) & b(M,2) & & & 1 \end{vmatrix} .$$

This is a convenient notation for representing the transformation from a set of correlated basis vectors $\underline{y}$ to an uncorrelated set of basis vectors $\underline{u}$. The bases $\underline{y}$ and $\underline{u}$ span the same M-dimensional subspace S, but there are no redundant correlations between the basis vectors of set $\underline{u}$. Since the basis of $\underline{u}$ is uncorrelated, its components u(i) (i = 1,2,...,M) are referred to as innovations because each additional component contributes completely new information to the estimate of $\underline{y}$. [Ref. 1:pp. 16-18]

Finally, the transformation $\underline{y} = B\underline{u}$ corresponds to a LU (lower upper)-Cholesky factorization of the correlation matrix of $\underline{y}$. By definition $R_{yy} = E[\underline{y}\underline{y}^T]$. Substituting for

$\underline{y}$ results in

$$R_{yy} = BE[\underline{u}\underline{u}^T]B^T = BR_{uu}B^T \qquad (2.58)$$

where B is lower triangular, B^T upper triangular, and R_{uu} is a diagonal matrix since the basis vectors u(i) are uncorrelated with one another.

3. Derivation of Lattice Filter Equations

Now the lattice filter order update equations will be developed. A random signal y(n) can be modeled as the output of a causal, stable, linear filter H(z) which is driven by a stationary, uncorrelated, white noise sequence {e(1),e(2),...,e(P)} [Ref. 1:p. 30]. A P-th order autoregressive model is defined by $H(z) = 1/A_P(z)$ where

$$A_P(z) = 1+a(1)z^{-1}+a(2)z^{-2}+\ldots+a(P)z^{-P} . \qquad (2.59)$$

The filter $A_P(z)$ has several names: prediction error filter, inverse filter, or analysis filter. The signal y(n) can be written as

$$y(n) = e(n) - \sum_{i=1}^{P} a(i)y(n-i) . \qquad (2.60)$$

If the predictor $\hat{y}(n)$ is introduced, this becomes

$$y(n) - \hat{y}(n) = e(n) \qquad (2.61)$$

where $\hat{y}(n) = -\sum_{i=1}^{P} a(i)y(n-i)$. Now, $\hat{y}(n)$ is the estimate of y(n) at time (n-1) based on the previous P samples, {y(n-P),

y(n-P+1),..., y(n-1)} and e(n) is equivalent to the estimation error. The estimate $\hat{y}(n)$ can be thought of as the projection of the (P+1)-dimensional y(n) vector onto the P-dimensional subspace spanned by the components of the data vector $Y = [y(n-1),y(n-2),...,y(n-P)]^T$. In general, the past values of y(n) are correlated with one another. Therefore, the random variable components of Y do not generally form a set of mutually orthogonal basis vectors. The Gram-Schmidt orthogonalization procedure removes these correlations and transforms Y into a set of mutually orthogonal basis vectors which span the same subspace. Additionally, the predictor coefficients a(i) are replaced by the reflection coefficients K_i. This results in the lattice filter.

In order to obtain the optimal estimator, the predictor coefficients which minimize the mean-squared prediction error $E[e^2(n)]$ must be determined. This problem was discussed in the introduction to this chapter. The resulting matrix form of the normal equations is

$$\begin{vmatrix} R(0) & R(1) & R(2) & \cdots & R(P) \\ R(1) & R(0) & R(1) & \cdots & \\ R(2) & R(1) & R(0) & \cdots & \\ \vdots & \vdots & \vdots & \ddots & \\ R(P) & & & & R(0) \end{vmatrix} \begin{vmatrix} 1 \\ a(1) \\ a(2) \\ \vdots \\ a(P) \end{vmatrix} = \begin{vmatrix} Ep \\ 0 \\ 0 \\ \vdots \\ 0 \end{vmatrix} \qquad (2.62)$$

where R(k) = E[y(n+k)y(n)] and E_P is the minimized mean-squared prediction error for the P-th order filter. The

sample autocorrelation matrix components are calculated from the length N data sequence by

$$R(k) = (1/N) \sum_{n=0}^{N-1-k} y(n+k)y(n) \qquad (2.63)$$

for $0 \leq k \leq P$ and where $P \leq N-1$. [Ref. 1:p. 150]

Now there are P+1 equations and P+1 unknown model parameters $\{a(1),a(2),\ldots,a(P);E_P\}$. The P+1 equations can be solved by inverting the autocorrelation matrix. This requires $O(P^3)$ operations and $O(P^2)$ storage locations. The P+1 equations can be solved more efficiently by taking advantage of the matrix's Toeplitz structure by using Levinson's algorithm. Levinson's algorithm reduces the required number of operations and storage locations to $O(P^2)$ and O(P), respectively [Ref. 1:pp. 150-151]. Being a recursive procedure, Levinson's algorithm permits the calculation of the (P+1)-st order model parameters by using the previously determined P-th order model parameters. The matrix form of the algorithm is given by

$$\begin{vmatrix} 1 \\ a_{P+1}(1) \\ a_{P+1}(2) \\ \vdots \\ a_{P+1}(P) \\ a_{P+1}(P+1) \end{vmatrix} = \begin{vmatrix} 1 \\ a_P(1) \\ a_P(2) \\ \vdots \\ a_P(P) \\ 0 \end{vmatrix} - K_{P+1} \begin{vmatrix} 0 \\ a_P(P) \\ a_P(P-1) \\ \vdots \\ a_P(1) \\ 1 \end{vmatrix} \qquad (2.64)$$

where

$$K_{P+1} = \frac{\sum_{i=0}^{P} a_P(i) R(P+1-i)}{\sum_{i=0}^{P} a_P(i) R(i)} \quad . \qquad (2.65)$$

The P subscript on the "a" parameters specifies the P-th order prediction error filter $A_P(z)$ whereas the index identifies the appropriate term in the $A_P(z)$ polynomial. K_{P+1} is called the (P+1)-st order reflection or PARCOR (partial correlation) coefficient. The PARCOR coefficient K_{P+1} represents the true, or direct, correlation between y(n-P-1) and y(n) with the effects of the intermediate variables (i.e., y(n-P),y(n-P+1),...,y(n-1)) removed. The recursive form of Levinson's algorithm is written as

$$a_{P+1}(m) = a_P(m) - K_{P+1}\, a_P(P+1-m) \text{ for } 1 \leq m \leq P, \qquad (2.66a)$$

and

$$a_{P+1}^{(P+1)} = -K_{P+1} \quad \text{for } m=P+1. \tag{2.66b}$$

This shows that the reflection coefficient for each stage P is equal to the highest coefficient of $A_P(z)$. There is a one-to-one correspondence between the PARCOR coefficients and the coefficients of the transfer function $A_P(z)$. The transfer function or, equivalently, the autocorrelation matrix $R_T = E[y(n)y^T(n)]$ uniquely determine the reflection coefficients [Ref. 7:p. 829]. Taking the z-transform of equation (2.66) results in

$$A_{P+1}(z) = A_P(z) - K_{P+1} B_P(z) \tag{2.67a}$$

where

$$A_P(z) = 1 + a_P^{(1)} z^{-1} + a_P^{(2)} z^{-2} + \ldots + a_P^{(P)} z^{-P}, \tag{2.67b}$$

and

$$B_P(z) = z^{-P} A_P(z^{-1}), \tag{2.67c}$$
$$= a_P^{(P)} + a_P^{(P-1)} z^{-1} + a_P^{(P-2)} z^{-2} + \ldots + a_P^{(1)} z^{-(P-1)} + z^{-P}.$$

Due to the effective folding about the axis and the shifting to the right of the sequence $A_P(z)$, the polynomial $B_P(z)$ is called the reverse of $A_P(z)$. Taking the reverse of both sides of equation (2.67) yields

$$B_{P+1}(z) = z^{-1} B_P(z) - K_{P+1} A_P(z) \tag{2.68}$$

Combining equations (2.67) and (2.68) into matrix form gives

$$\begin{vmatrix} A_{P+1}(z) \\ B_{P+1}(z) \end{vmatrix} = \begin{vmatrix} 1 & -K_{P+1} z^{-1} \\ -K_{P+1} & z^{-1} \end{vmatrix} \begin{vmatrix} A_P(z) \\ B_P(z) \end{vmatrix} . \quad (2.69)$$

The forward prediction error associated with predicting $y(n)$ from the previous P samples $\{y(n-P),\ldots y(n-1)\}$ is written as $e_P(n) = y(n) - \hat{y}_P(n)$ where the subscript P represents the filter order. In z-domain notation, this becomes

$$E_P(z) = A_P(z)Y(z) \quad (2.70)$$

Now, the backward prediction error $r_P(n)$ is introduced. It is defined as the error in predicting (or actually smoothing) $y(n-P)$ from the future P samples $\{y(n-P+1,\ldots, y(n)\}$. It is written as

$$r_P(n) = y(n-P)+a_P(1)y(n-P+1)+\ldots+a_P(P)y(n) \quad (2.71a)$$

or in z-domain notation as

$$\tilde{E}_P(z) = B_P(z)Y(z) \quad . \quad (2.71b)$$

The optimal backward predictor coefficients minimize the mean-squared smoothing error $E[r_P^2(n)]$. Since the optimal backward and forward predictor coefficients are mirror

images of each other, then $E[r^2_p(n)] = E[e^2_p(n)]$. That is, the backward and forward prediction error vectors have the same norms [Ref. 8:p. 101].

As will be shown, at each instant n, the backward prediction errors are mutually orthogonal (i.e., uncorrelated if assumed to be zero mean) [Ref. 1:p. 170]. Therefore, it is the backward prediction errors, $r_p(n), p=0,1,\ldots,M$, where M is the filter order, which form the new set of basis vectors. To demonstrate that the backward prediction errors are mutually orthogonal, let M=3 and write the corresponding equations for P = 0,1,2,3 in matrix form:

$$\begin{vmatrix} r_0(n) \\ r_1(n) \\ r_2(n) \\ r_3(n) \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 & 0 \\ a_1(1) & 1 & 0 & 0 \\ a_2(2) & a_2(1) & 1 & 0 \\ a_3(3) & a_3(2) & a_3(1) & 0 \end{vmatrix} \begin{vmatrix} y(n) \\ y(n-1) \\ y(n-2) \\ y(n-3) \end{vmatrix} , \qquad (2.72a)$$

or

$$\underline{r}(n) = L\underline{y}(n) \quad . \qquad (2.72b)$$

Note that the first column of L contains the negatives of all the reflection coefficients. Now examine the covariance matrix of $\underline{r}(n)$:

$$R_{rr} = E[\underline{r}(n)\underline{r}^T(n)] = E[L\underline{y}(n)(L\underline{y}(n))^T] \qquad (2.73)$$
$$= LE[\underline{y}(n)\underline{y}^T(n)]L^T = LR_{yy}L^T .$$

Since the normal equations are satisfied within the matrix products above, R_{rr} reduces to a diagonal matrix which verifies that the components of $\underline{r}(n)$ are uncorrelated. That is

$$R_{rr} = LR_{yy}L^T = D = \text{diag}\{E_0, E_1, E_2, E_3\} \qquad (2.74)$$

where E_P is the minimum value of the mean-squared prediction error which is given by $E[e_P^2(n)]$. It can be shown that $E_{P+1} = (1-K_{P+1}^2)E_P$ where the recursion is initialized with the value given by $E_0 = R_{yy}(0) = E[y^2(n)]$ [Ref. 8:p. 105]. Therefore, the prediction error decreases by a factor of $(1-K_{P+1}^2)$ from one lattice stage to the next. Now, since the elements of $\underline{r}(n)$ are uncorrelated, equation (2.72) is equivalent to the Gram-Schmidt orthogonalization of the data vector $\underline{y}(n)$. Also note that rewriting equation (2.74) as $R_{yy} = L^{-1}DL^{-T}$ corresponds to a LU-Cholesky factorization of the covariance matrix of $\underline{y}(n)$.

To complete the derivation of the lattice recursion equations, multiply both sides of equation (2.69) by Y(z) to get

$$\begin{vmatrix} E_{P+1}(z) \\ \tilde{E}_{P+1}(z) \end{vmatrix} = \begin{vmatrix} 1 & -K_{P+1} z^{-1} \\ -K_{P+1} & z^{-1} \end{vmatrix} \begin{vmatrix} E_P(z) \\ \tilde{E}_P(z) \end{vmatrix} . \quad (2.75)$$

These equations are transformed into the time domain to obtain the final result:

$$e_{P+1}(n) = e_P(n) - K_{P+1} r_P(n-1) \quad (2.76a)$$

$$r_{P+1}(n) = r_P(n-1) - K_{P+1} e_P(n) . \quad (2.76b)$$

These equations, which are recursive in both time and order, define the signal flow graph of the analysis lattice filter, shown in Figure 2.6. For a given time instant n, the equations are evaluated recursively in order. The inputs into the first stage of the lattice filter are $e_0(n) = r_0(n) = y(n)$. The successive stages of the filter develop the successively higher order forward and backward prediction errors. The output from the final stage yields the desired M-th order forward prediction error, while all lower order prediction errors are available at intermediate stages. Since the backward errors are generated from the $\underline{y}$ data vector, the analysis lattice filter actually implements the Gram-Schmidt orthogonalization; all that is required to implement the lattice are the reflection factors $\{K_1, K_2, \ldots, K_M\}$.

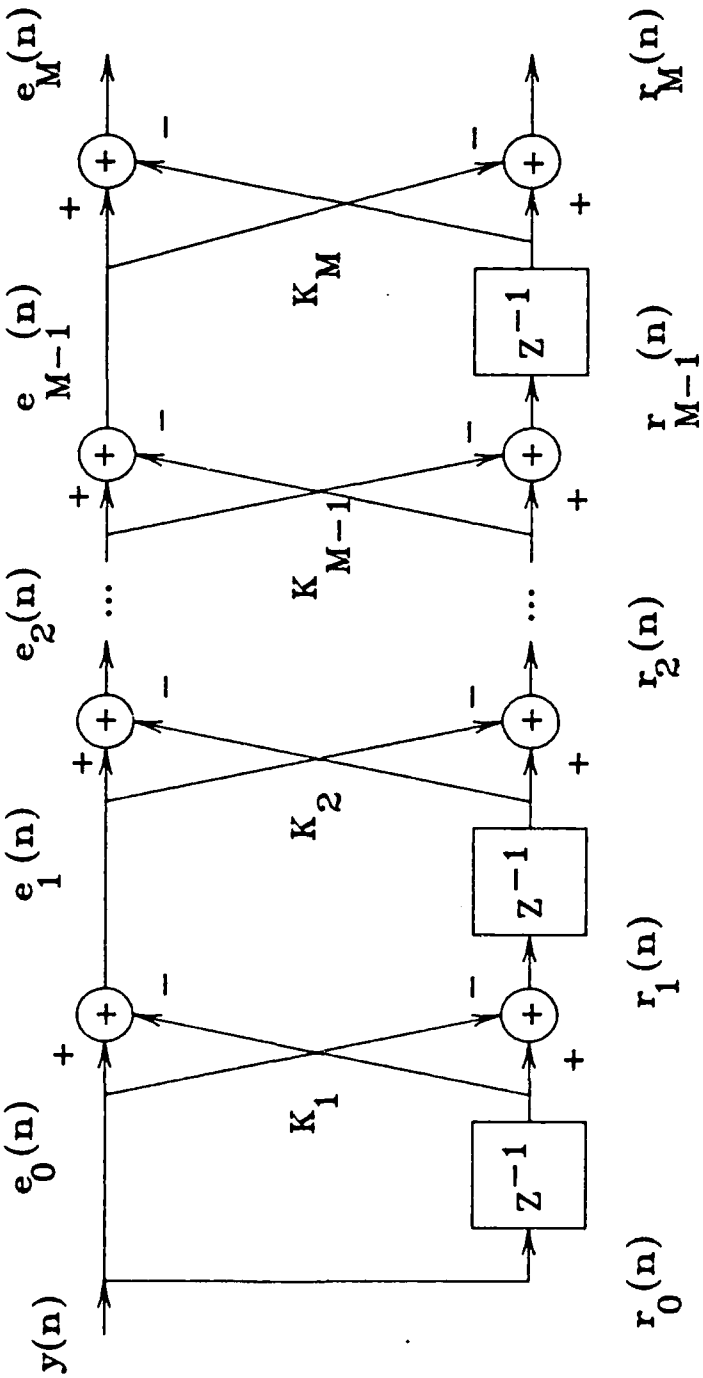


Figure 2.6 Analysis Lattice Filter

Equation (2.76) can be manipulated to obtain the lattice form of the synthesis filter $H(z) = 1/A_M(z)$:

$$e_P(n) = e_{P+1}(n) + K_{P+1}\, r_P(n-1) \qquad (2.77a)$$

$$r_{P+1}(n) = r_P(n-1) - K_{P+1}\, e_P(n) \,. \qquad (2.77b)$$

Figure 2.7 shows the corresponding signal flow graph. When the input to the synthesis filter is the forward prediction error sequence $e_P(n)$, the output is the original sequence $y(n)$. In order for the synthesis filter to be stable and causal, all M zeros of the prediction-error filter $A_M(z)$ must lie within the unit circle in the complex z-plane. A necessary and sufficient condition for all the zeros to be inside the unit circle is that the magnitude of each of the reflection coefficients $\{K_1, K_2, \ldots, K_M\}$ be less than one. [Ref. 1:pp. 168-169]

The reflection factors can be evaluated by several methods. The various methods arise due to different definitions of the optimality criterion. The criterion used here of minimizing the mean-squared prediction errors leads to what MAKHOUL calls the forward and backward methods [Ref. 9]. They are, respectively:

$$K_{P+1,e} = E[e_P(n) r_P(n-1)] \,/\, E[r_P^2(n-1)] \qquad (2.78a)$$

$$K_{P+1,r} = E[e_P(n) r_P(n-1)] \,/\, E[e_P^2(n)] \,. \qquad (2.78b)$$

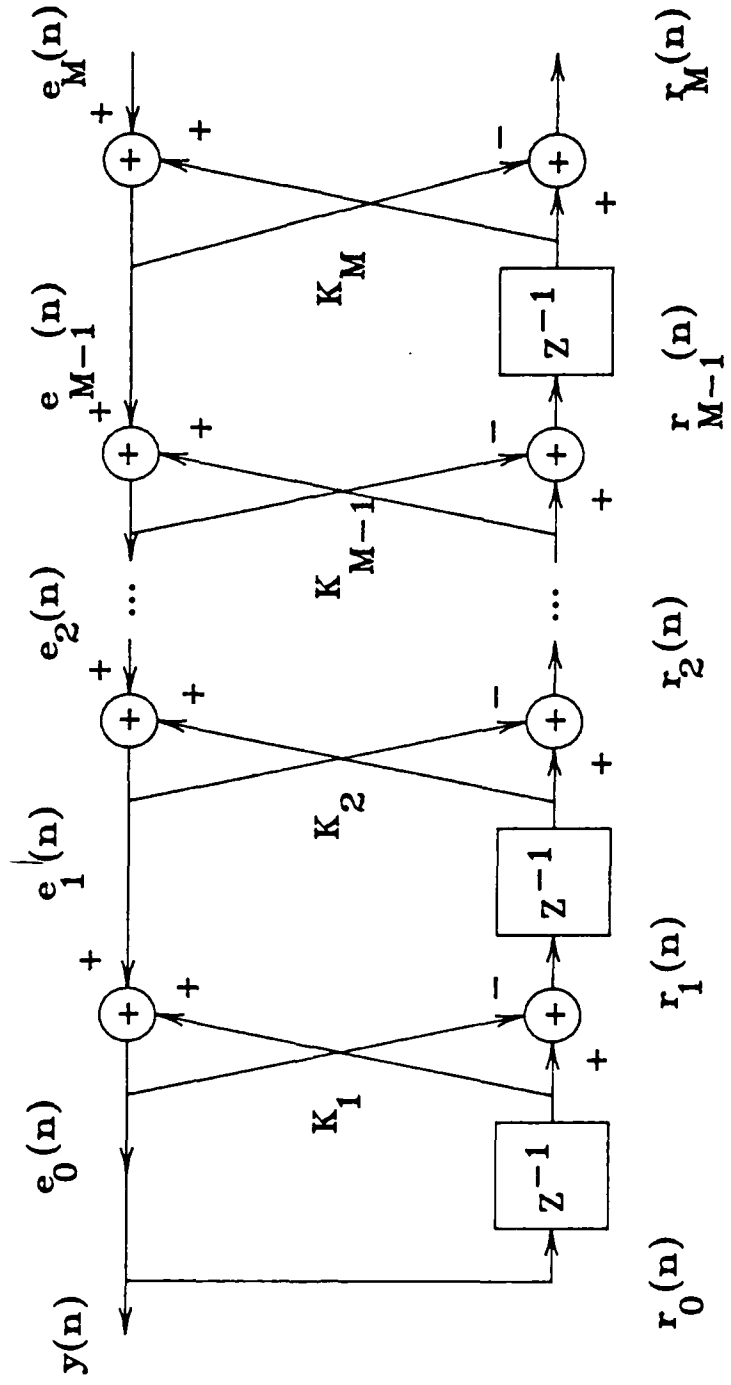


Figure 2.7 Synthesis Lattice Filter

Assuming stationarity, and since $E[r_P^2(n)] = E[e_P^2(n)]$ as previously discussed, then the forward and backward reflection coefficients are equal. That is $K_{P+1} = K_{P+1,e} = K_{P+1,r}$. The Schwarz inequality implies that $|K_P| \leq 1$ for each P=1,2,...,M [Ref. 8:p. 104]. An alternate technique for calculating the reflection coefficients is the geometric mean method. It was introduced by ITAKURA and SAITO in their work of developing a digital filter structure for time-domain speech analysis [Ref. 10]. The corresponding PARCOR coefficient is given by

$$K_{P+1} = \frac{E[e_P(n)r_P(n-1)]}{\{E[e_P^2(n)]E[r_P^2(n-1)]\}^{1/2}} \qquad (2.79)$$

These PARCOR coefficients are guarenteed to have magnitudes less than one [Ref. 7:p. 840].

A third method was used by BURG in the maximum-entropy method of spectral estimation [Ref. 11]. This is the harmonic-mean method. The harmonic-mean method seeks to minimize the sum of the forward and backward prediction error variances, $E[e_P^2(n)] + E[r_P^2(n-1)]$. Minimizing this sum with respect to the reflection coefficients results in

$$K_{P+1} = \frac{2E[e_P(n)r_P(n-1)]}{E[e_P^2(n)] + E[r_p^2(n-1)]} \qquad (2.80)$$

Again, the Schwarz inequality verifies that K_P has magnitude less than one, guaranteeing that the synthesis filter is causal and stable [Ref. 1:p. 189].

4. The Generalized (Analysis) Lattice Filter

The preceding development of the analysis lattice filter equations assumed that the data sequence {y(n)} represented a time sequence with stationary statistics. In this section, a more general linear prediction problem will be considered. No special assumptions are made concerning the data. The data values need not be delayed versions of each other; they need not even represent a time sequence. This is the approach taken by LENK in developing the generalized order update equations [Ref. 12:p. 85]. The resulting generalized form of the lattice filter makes it suitable for multidimensional and nonlinear signal processing applications.

Definitions associated with the normalized form of the generalized lattice filter will now be introduced. First, the components of the length (M+1) column vector $[y^{\lambda}]$, representing a single realization of the random process Y, are designated by y^{λ}, $\lambda = 0,1,\ldots,M$. The forward prediction error in estimating the (n+1)-st element from

the previous N elements of the data vector is written as

$$e^{N}_{n+1} = \sum_{\lambda=0}^{M} h^{N}_{\lambda}(n+1)y^{\lambda} \qquad (2.81)$$

where the length (M+1) row vector of coefficients is given by

$$[h^{N}_{\lambda}(n+1)]=[0,\ldots,0,-h^{N}_{n-N+1},-h^{N}_{n-N+2},\ldots,-h^{N}_{n},1,0,\ldots,0]. \qquad (2.82)$$

The backwards prediction error associated with predicting y^{n-N} from the next N elements of the data vector is given by

$$r^{N}_{n-N} = \sum_{\lambda=0}^{M} \tilde{h}^{N}_{\lambda}(n-N)y^{\lambda} \qquad (2.83)$$

where the associated coefficients are given by the vector

$$[\tilde{h}^{N}_{\lambda}(n-N)]=[0,\ldots 0,1,-\tilde{h}^{N}_{n-N+1},-\tilde{h}^{N}_{n-N+2},\ldots,-\tilde{h}^{N}_{n},0,\ldots,0]. \qquad (2.84)$$

The norm of the forward prediction error is defined as

$$||e^{N}_{n+1}|| = [E\{(e^{N}_{n+1})^{2}\}]^{1/2}. \qquad (2.85)$$

The norm of the backward prediction error is defined in a similar manner. Now, the normalized forward and backward N-th order prediction errors are defined by

$$\vec{e}^{N}_{n+1} = e^{N}_{n+1} / ||e^{N}_{n+1}|| = \sum_{\lambda=0}^{M} a^{N}_{\lambda}(n+1)y^{\lambda}, \qquad (2.86a)$$

and

$$\vec{r}^{N}_{n-N} = r^{N}_{n-N} / ||r^{N}_{n-N}|| = \sum_{\lambda=0}^{M} b^{N}_{\lambda}(n-N)y^{\lambda} \qquad (2.86b)$$

where the normalized prediction error coefficients are defined as

$$a_{\lambda}^{N}(n+1) = h_{\lambda}^{N}(n+1) \; / \; \| e_{n+1}^{N} \| \qquad (2.87a)$$

and

$$b_{\lambda}^{N}(n-N) = \tilde{h}_{\lambda}^{N}(n-N) \; / \; \| r_{n-N}^{N} \| \; . \qquad (2.87b)$$

[Ref. 12:pp. 85-87]

Using the normalized form of the generalized Levinson algorithm, LENK demonstrated that equations (2.87) could be updated recursively through the relation

$$\begin{vmatrix} a_{\lambda}^{N+1}(n+1) \\ b_{\lambda}^{N+1}(n-N) \end{vmatrix} = \theta(K_{N+1}^{n}) \begin{vmatrix} a_{\lambda}^{N}(n+1) \\ b_{\lambda}^{N}(n-N) \end{vmatrix} \qquad (2.88)$$

where the partial correlation (PARCOR) coefficient is

$$K_{N+1}^{n} = E\{\bar{e}_{n+1}^{N} \, \bar{r}_{n-N}^{N}\} \; , \qquad (2.89)$$

and where

$$\theta(K_{N+1}^{n}) = \frac{1}{1 - (K_{N+1}^{n})^{2}} \begin{vmatrix} 1 & -K_{N+1}^{n} \\ -K_{N+1}^{n} & 1 \end{vmatrix} . \qquad (2.90)$$

The recursion is started for order N=0 with the initial prediction error values for $\lambda = 0, 1, \ldots, M$ given by the vectors

$$[a_{\lambda}^{0}(n+1)] = [0,\ldots,0,\ 1/\|y^{n+1}\|\ ,0,\ldots,0] \qquad (2.91a)$$

$$[b_{\lambda}^{0}(n)] = [0,\ldots,0,\ 1/\|y^{n}\|,\ 0,\ldots,0]\ . \qquad (2.91b)$$

Multiplying both sides of equation (2.90) by the data vector $[y^{\lambda}]$ yields the desired error order update equations:

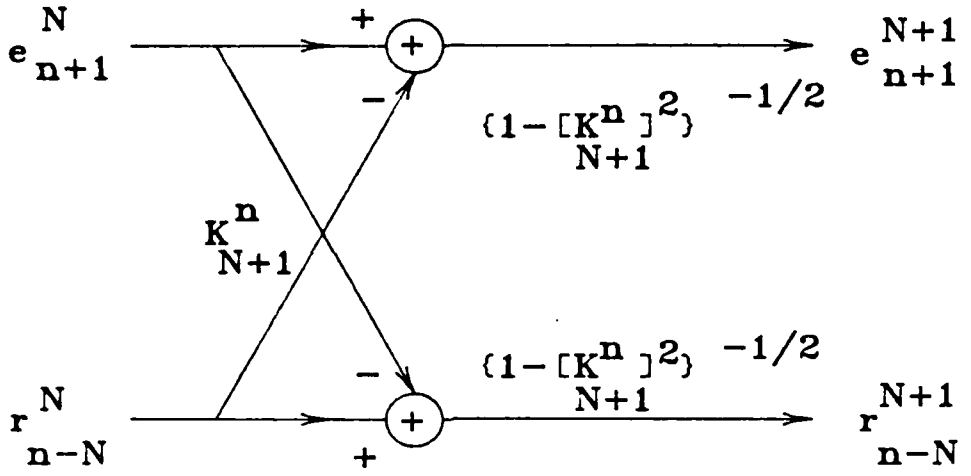
$$\begin{vmatrix} \bar{e}_{n+1}^{N+1} \\ \bar{r}_{n-N}^{N+1} \end{vmatrix} = \theta(K_{N+1}^{n}) \begin{vmatrix} \bar{e}_{n+1}^{N} \\ \bar{r}_{n-N}^{N} \end{vmatrix} \qquad (2.92)$$

The corresponding signal flow graph for a single lattice filter section is shown in Figure 2.8. Figure 2.9 depicts a third order generalized lattice filter. [Ref. 12:pp. 94-99]

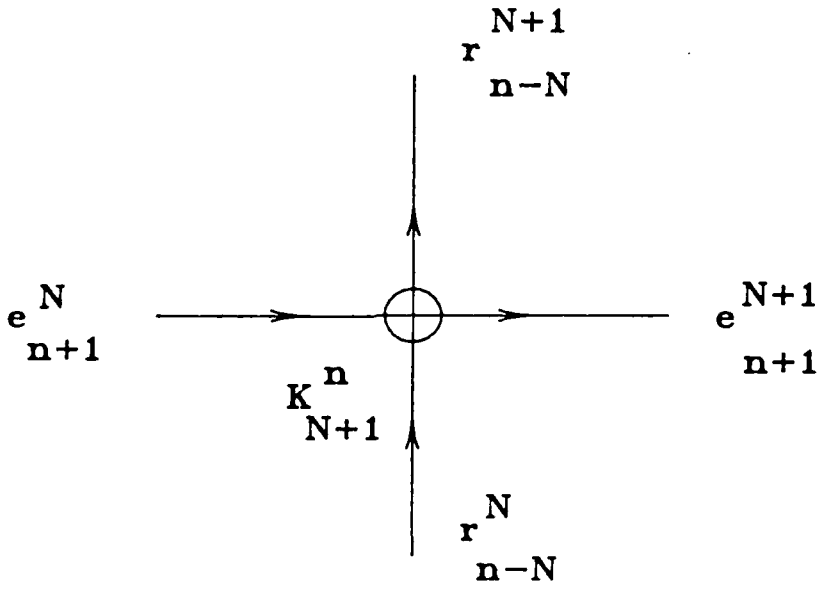
LENK then proved that the reflection coefficients (i.e., PARCOR coefficients) could be calculated directly from the data vector's correlation matrix by utilizing the generalized Schur algorithm. In LENK's notation, the components of the correlation matrix are written as $R^{\mu\lambda} = E\{y^{\mu}\ y^{\lambda}\}$. Then the reflection coefficients are given by

$$K_{N+1}^{n} = \frac{\alpha_{N}^{n-N}(n+1)}{\beta_{N}^{n-N}(n-N)}\ , \qquad (2.93)$$

where



a. Single Stage of Generalized Lattice Filter



b. Alternate Representation

Figure 2.8 Generalized Lattice Filter

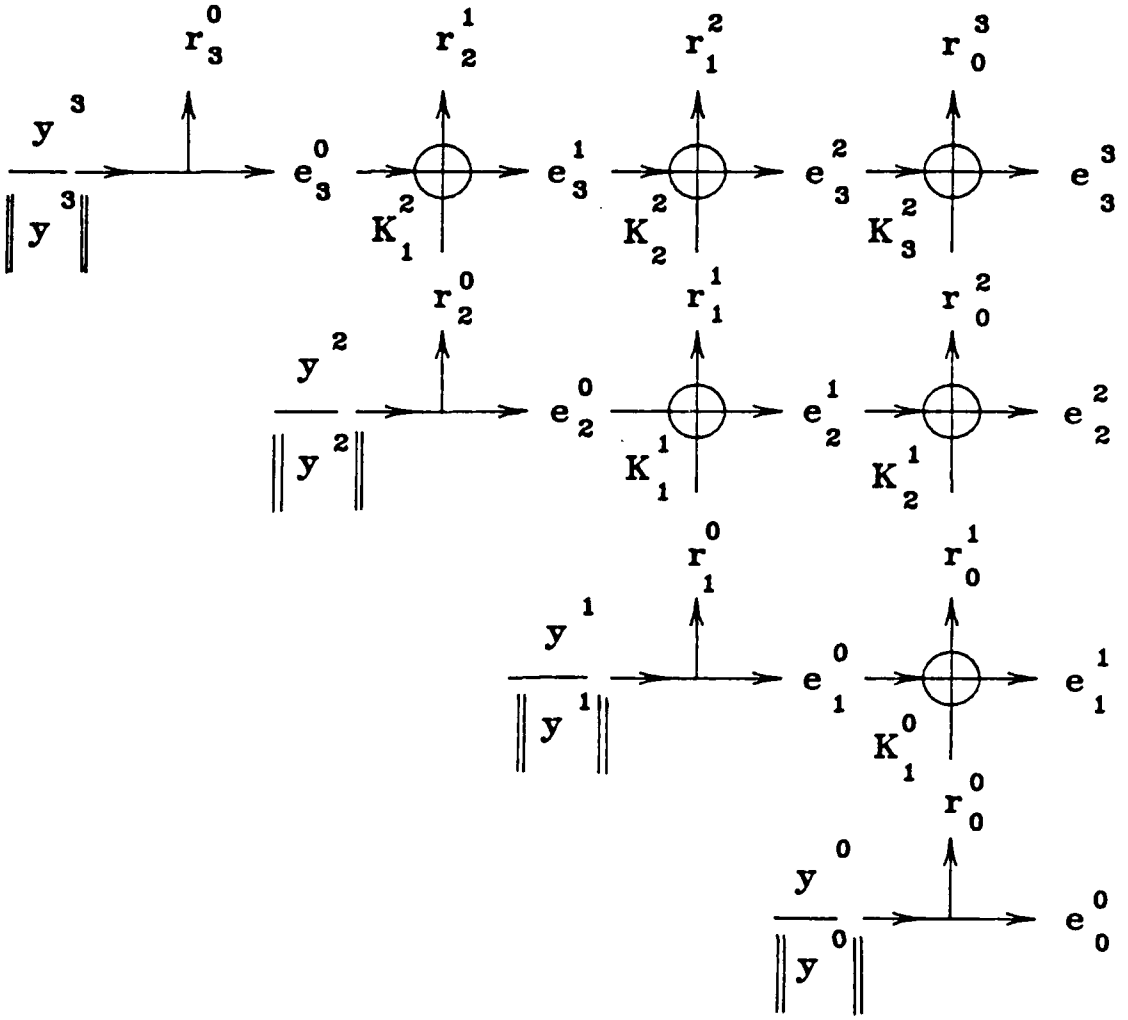


Figure 2.9 3–rd Order Generalized Lattice Filter

$$\alpha_{N+1}^{\lambda}(n+1) = E\{\bar{e}_{n+1}^{N} y^{\lambda}\} = \sum_{\mu=0}^{M} a^{N}(n+1) R^{\mu\lambda} , \qquad (2.94a)$$

and

$$\beta_{N+1}^{\lambda}(n-N) = E\{\bar{r}_{n-N}^{N} y^{\lambda}\} = \sum_{\mu=0}^{M} b^{N}(n-N) R^{\mu\lambda} . \qquad (2.94b)$$

Equations (2.94) can be updated through the recursion relation

$$\begin{vmatrix} \alpha_{N+1}^{\lambda}(n+1) \\ \beta_{N+1}^{\lambda}(n-N) \end{vmatrix} = \begin{pmatrix} n \\ K_{N+1} \end{pmatrix} \begin{vmatrix} \alpha_{N}^{\lambda}(n+1) \\ \beta_{N}^{\lambda}(n-N) \end{vmatrix} \qquad (2.95)$$

To start the recursion at order N=0, the values of the parameters for $\lambda = 0,1,\ldots,M$ are given by the vectors

$$[\alpha_{0}^{\lambda}(n+1)] = \|y^{n+1}\|^{-1} [R^{(n+1)0}, R^{(n+1)1}, \ldots, R^{(n+1)M}] \qquad (2.96a)$$

$$[\beta_{0}^{\lambda}(n)] = \|y^{n}\|^{-1} [R^{n0}, R^{n1}, \ldots, R^{nM}] . \qquad (2.96b)$$

[Ref. 12:pp. 100-101]

5. Lattice Filter Advantages

The lattice filter has several advantages compared to the direct, or transversal, form of the prediction error filter $A_M(z)$. First of all, due to the built-in orthogonalization incorporated into the lattice, successive lattice stages are decoupled. Therefore, the reflection

coefficients K_P, P=1,2,...,M are independent of the filter's final order. The M-th order least squares prediction filter contains all the prediction error filters of lower order. Restated, the first P sections of the M-th order filter form the P-th order prediction filter. Therefore, lattice stages may be added or subtracted from the existing lattice filter without having to recalculate the already determined reflection coefficients. In contrast, when the order of the transversal filter is changed, all the $A_M(z)$ prediction error filter coefficients must be recalculated. As a consequence, to specify all filter orders up to M requires storage of M(M+1)/2 predictor coefficients, while only M reflection coefficients need to be stored. [Ref. 7:p. 830]

Another advantage of the lattice over the transversal filter is that the output of each lattice stage is a least-squares prediction error. Therefore, if the desired order of the predictor is not known in advance, the output of the various stages can be monitored to determine what filter order is adequate. [Ref. 8:p. 106]

Due to the quantization inherent in implementing digital filters, round off noise is introduced. Studies have shown that the performance of the lattice filter is far superior to that of transversl filters for finite word length computations. Furthermore, the filter zeros are less sensitive to quantization errors in the reflection

coefficients than for quantization errors in the transversal filter coefficients. Finally, since the magnitude of the reflection coefficients is less than one, it simplifies the task of establishing the quantizer overload point. [Ref. 8:p. 106]

Another advantage the lattice filter holds over the transveral filter is that the lattice filter is minimum phase (i.e., all its zeros are inside the unit circle so that the synthesis model is stable) if and only if the magnitude of the reflection coefficients is less than one. There is no comparable test for the transversal filter coefficients to determine whether the synthesis model is stable. [Ref. 8:p. 106]

6. Linear Lattice Filter Applied to Deconvolution

The transfer function of the lattice filter is determined by the reflection coefficients. In turn, the values of these reflection coefficients are uniquely determined by the prediction error filter transfer function $A_M(z)$, or equivalently by the autocorrelation sequence $R(k)$, $k = 0,1,....M$ [Ref. 7:p. 829]. Similarly, equations (2.70) and (2.71b) indicate that the transfer functions from the input $y(n)$ to the outputs $e_M(n)$ and $r_M(n)$ are $A_M(z)$ and $B_M(z)$, respectively. Therefore, the analysis lattice filter is equivalent to the whitening filter $A_M(z)$; that is, it is the inverse of the system model $H(z) = 1/A_M(z)$. This is the

basis for using the lattice filter in a deconvolution application.

In order to recover the input signal x(n) from the system's output signal y(n), it is necessary to determine the inverse of the system transfer function H(z). The initial step is to conduct an "identification experiment" to estimate the system's parameters (either the autoregressive filter coefficients or the lattice filter reflection coefficients). To accurately estimate the parameters, the chosen experimental input signal must be sufficiently rich in frequency content, or more formally, persistently exciting so that it excites all the modes of the system. A sequence is said to be persistently exciting of order n if its (n x n) autocorrelation matrix is nonsingular [Ref. 13:pp. 70-71]. By using the techniques presented in section D.3 of this chapter, the resulting output sequence y(n) can be processed to yield the desired reflection coefficients. When the analysis lattice filter is implemented with these coefficients embedded in its structure, the lattice becomes the inverse of $A_M(z) = 1/H(z)$. This procedure is depicted in Figure 2.10.

The application of linear lattice filters to deconvolution was simulated using the FORTRAN programs LININV, ATOCOR, LEVIN, AND LATICE. These programs are provided in Appendix A. The simulation was conducted for a second

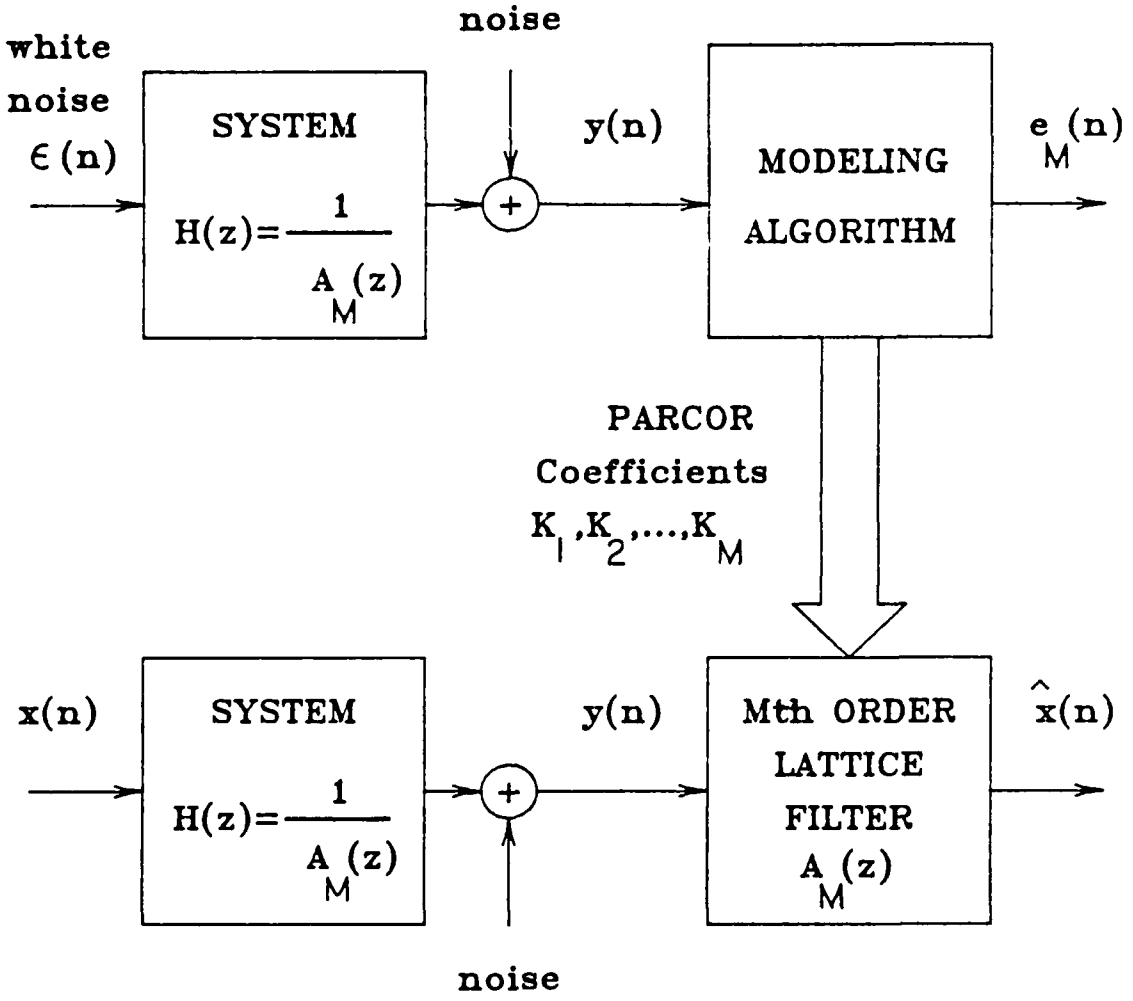


Figure 2.10 Deconvolution Using Lattice Filter

order, autoregressive, LTI system defined by the equation $y(n) = x(n) - (0.6y(n-1) + 0.08y(n-2))$. As previously mentioned, modeling the system was the first step in the deconvolution process. In order to identify the "unknown" parameters, the system was excited by a zero mean, unity variance, Gaussian white noise sequence. The output sequence y(n) was processed by subroutine ATOCOR to calculate the components of the autocorrelation matrix R_{yy} . Then subroutine LEVIN implemented Levinson's algorithm to evaluate both the prediction error filter coefficients and the associated reflection coefficients from the autocorrelation matrix. The actual output of subroutine LEVIN is the lower triangular L matrix described in equation (2.72); the i-th row of L contains the i-th order prediction error filter coefficients listed in reverse order, and the first column contains the negative of the reflection coefficients. Once the reflection coefficients corresponding to 1/H(z) were calculated, they were embedded in subroutine LATICE which implemented the analysis lattice filter equations (2.76). With the inverse filter of H(z) now available (i.e., the analysis lattice filter), H(z) was driven by an "unknown" signal x(n). The output of H(z) was then fed into the lattice filter. An approximation to x(n) was recovered at the forward error output signal from the last stage of the lattice.

Simulations were run both with and without the presence of measurement noise. Table 2.1 displays the resulting L matrices for one simulation. The case of no measurement noise is shown in Figures 2.11 through 2.14. Figure 2.11 is a plot of the input signal x(n), Figure 2.12 depicts the system output y(n), and Figure 2.13 shows the lattice filter output $\hat{x}(n)$. As can be seen, the results of the inverse filtering were excellent--the x and $\hat{x}$ curves are identical. This is verified by Figure 2.14 which shows that the mean-square error between x(n) and $\hat{x}(n)$ is nearly zero. The running average mean-square error was calculated using the equation

$$MSE(n) = \sqrt{(1/n) \sum_{i=1}^{n} (x(i) - \hat{x}(i))^2} \quad . \qquad (2.97)$$

The simulation was then repeated for a nonzero measurement noise. Here, the added measurement noise was a zero mean, 0.0025 variance, Gaussian white noise sequence. Using the same input as for the previous case, the outputs of the system and lattice filter are shown in Figures 2.15 and 2.16, respectively. Figure 2.17 is a plot of the mean-square error between x(n) and $\hat{x}(n)$. The results in Table 2.1 show that even with the presence of measurement noise, the system parameters were still accurately identified. The lattice filter recovered the basic shape of x(n), however, it could not remove the additive measurement noise.

Therefore, the output of the lattice filter is a noisy or "fuzzy" version of x(n). Additional filtering is required to remove the noise component of $\hat{x}(n)$. The results presented here are representative of those obtained for simulations involving other stable, autoregressive, LTI systems.

TABLE 2.1

RESULTS OF MODELING A LINEAR SYSTEM

Modeling Problem:

System: $H(z) = 1/[1 + 0.6z^{-1} + .08z^{-2}]$

System input: Gaussian white noise, N(0,1)

Number of white noise realizations used: 25

Number of points per realization: 5,000

Modeling Results:

a. No Measurement Noise

$$L = \begin{vmatrix} 1 & 0 & 0 \\ 0.555 & 1.0 & 0 \\ 0.077 & 0.598 & 1 \end{vmatrix}$$

Order,P	Ep	Kp
0	1.445	--
1	0.999	-0.555
2	0.993	-0.077

b. Measurement Noise is N(0,0.0025)

$$L = \begin{vmatrix} 1 & 0 & 0 \\ 0.555 & 1.0 & 0 \\ 0.077 & 0.598 & 1 \end{vmatrix}$$

Order,P	Ep	Kp
0	1.449	--
1	1.002	-0.555
2	0.996	-0.077

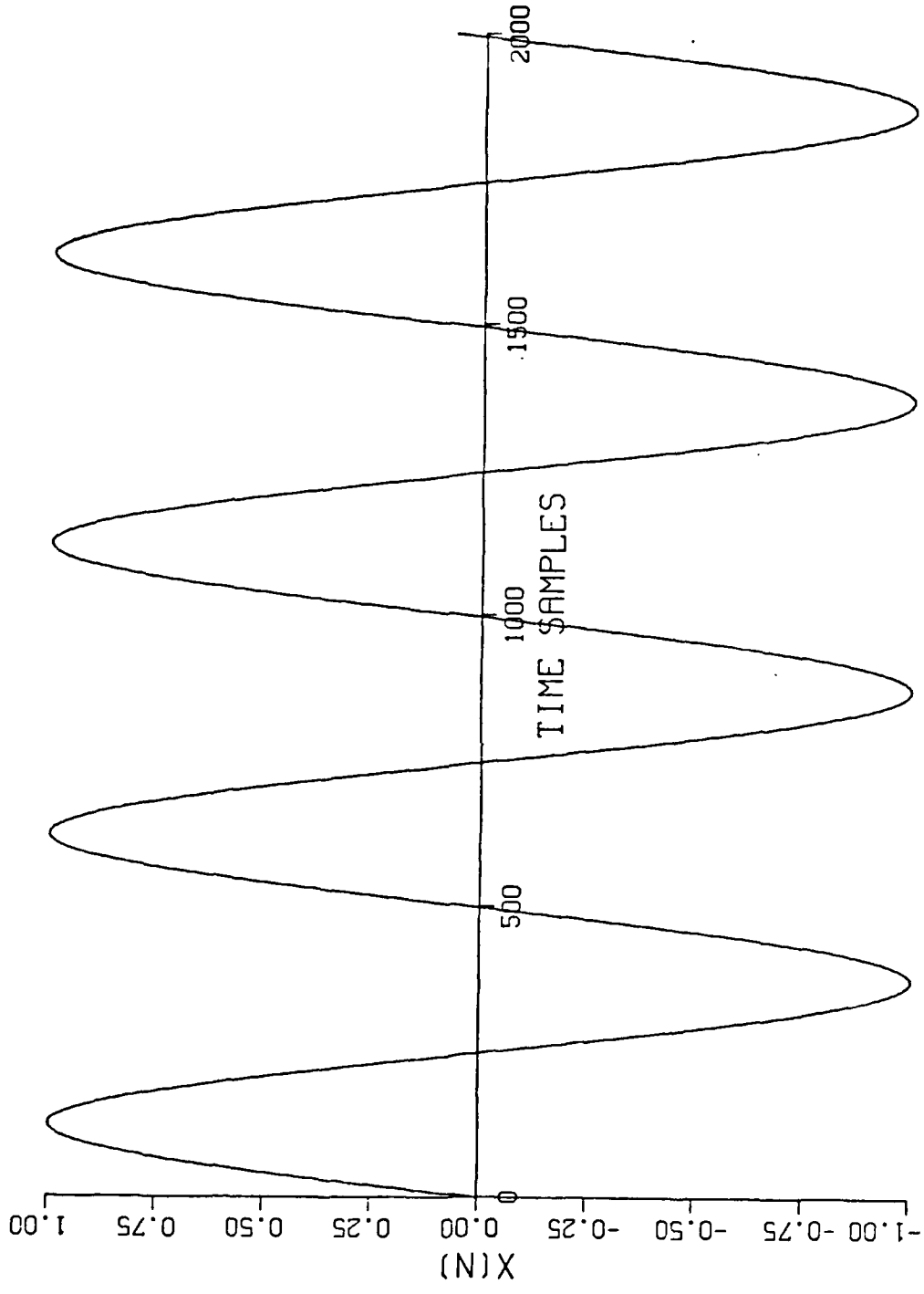


Figure 2.11 System Input, x(n)

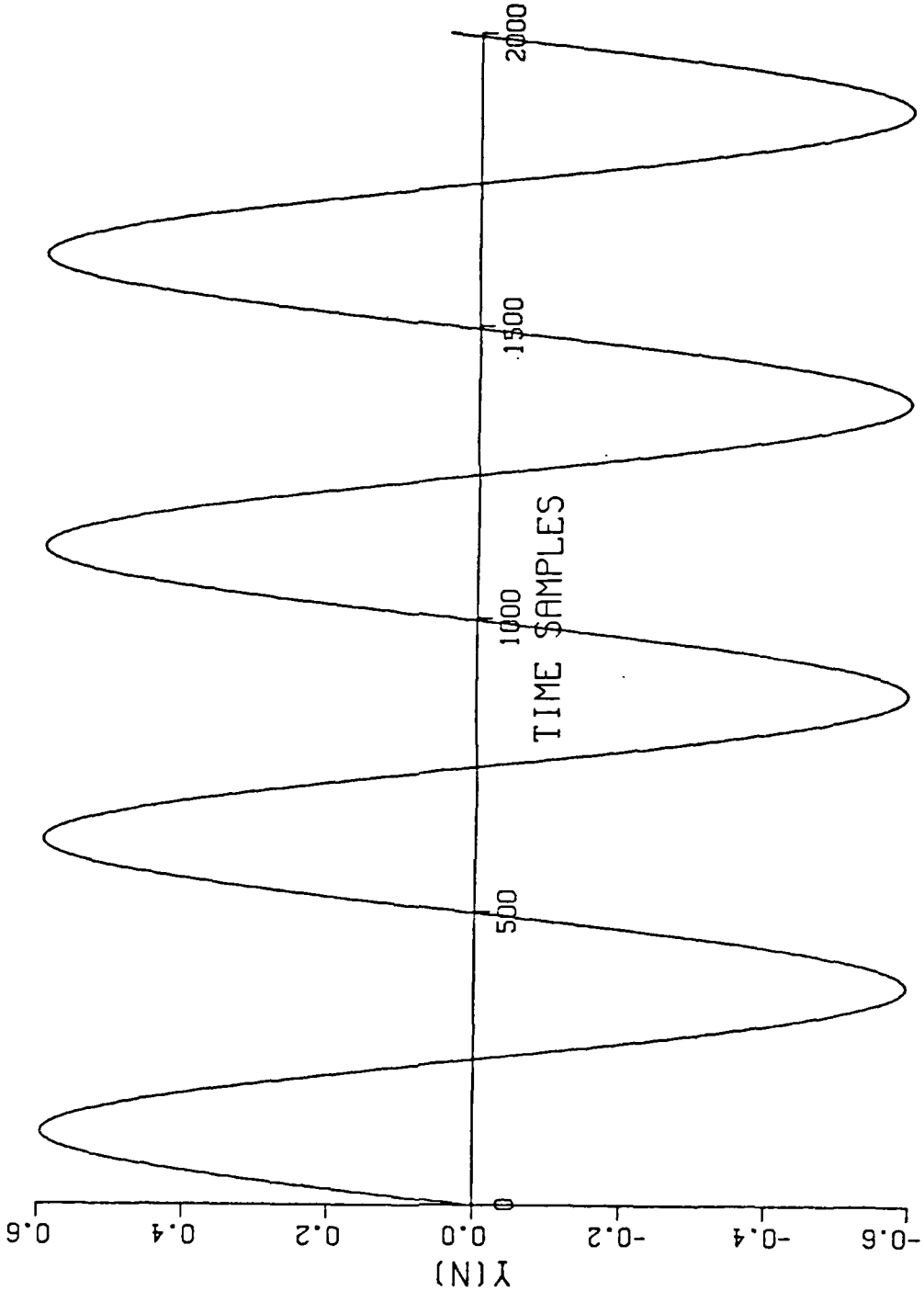


Figure 2.12 System Output, y(n)

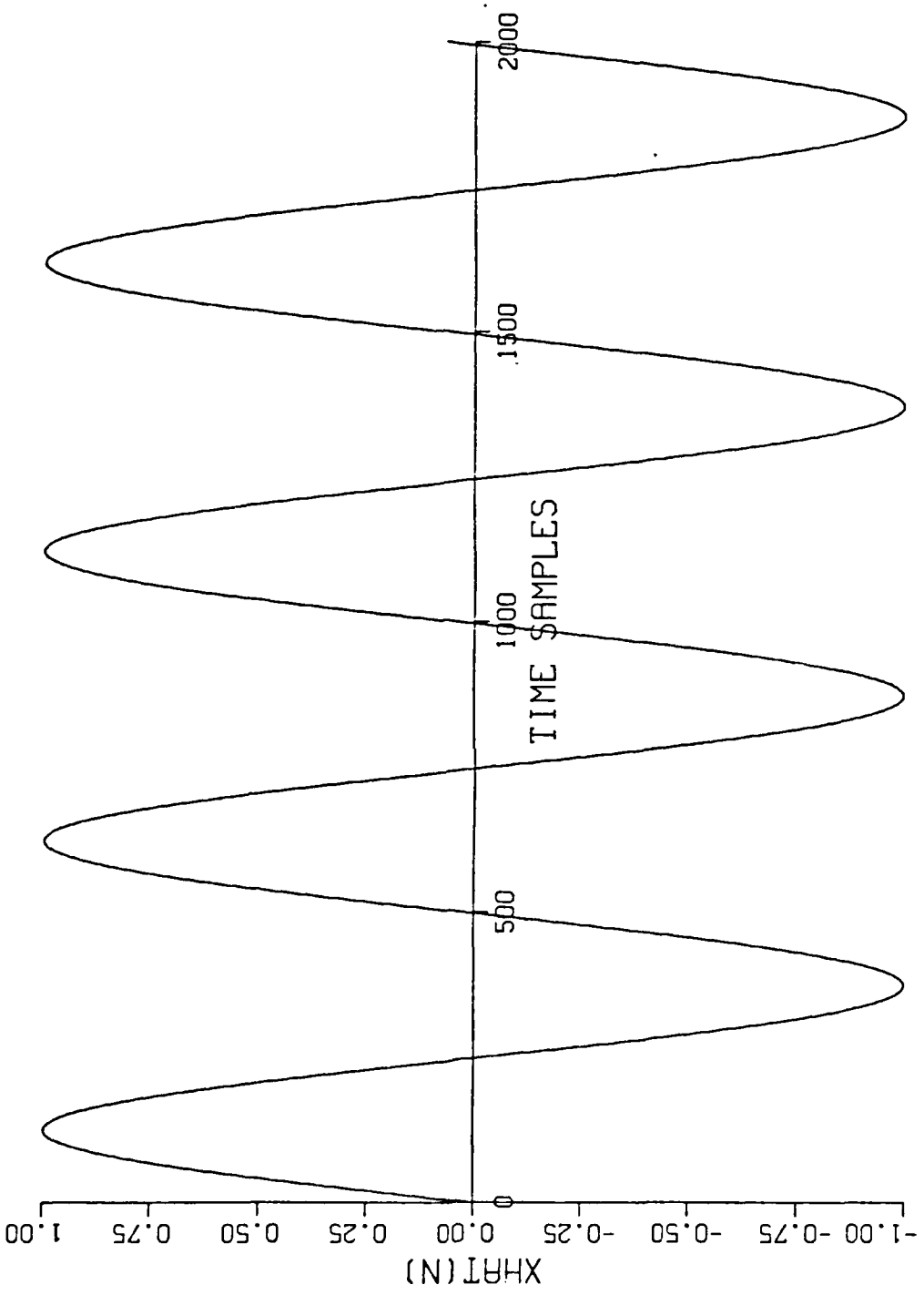


Figure 2.13 Lattice Filter Output, $\hat{x}(n)$

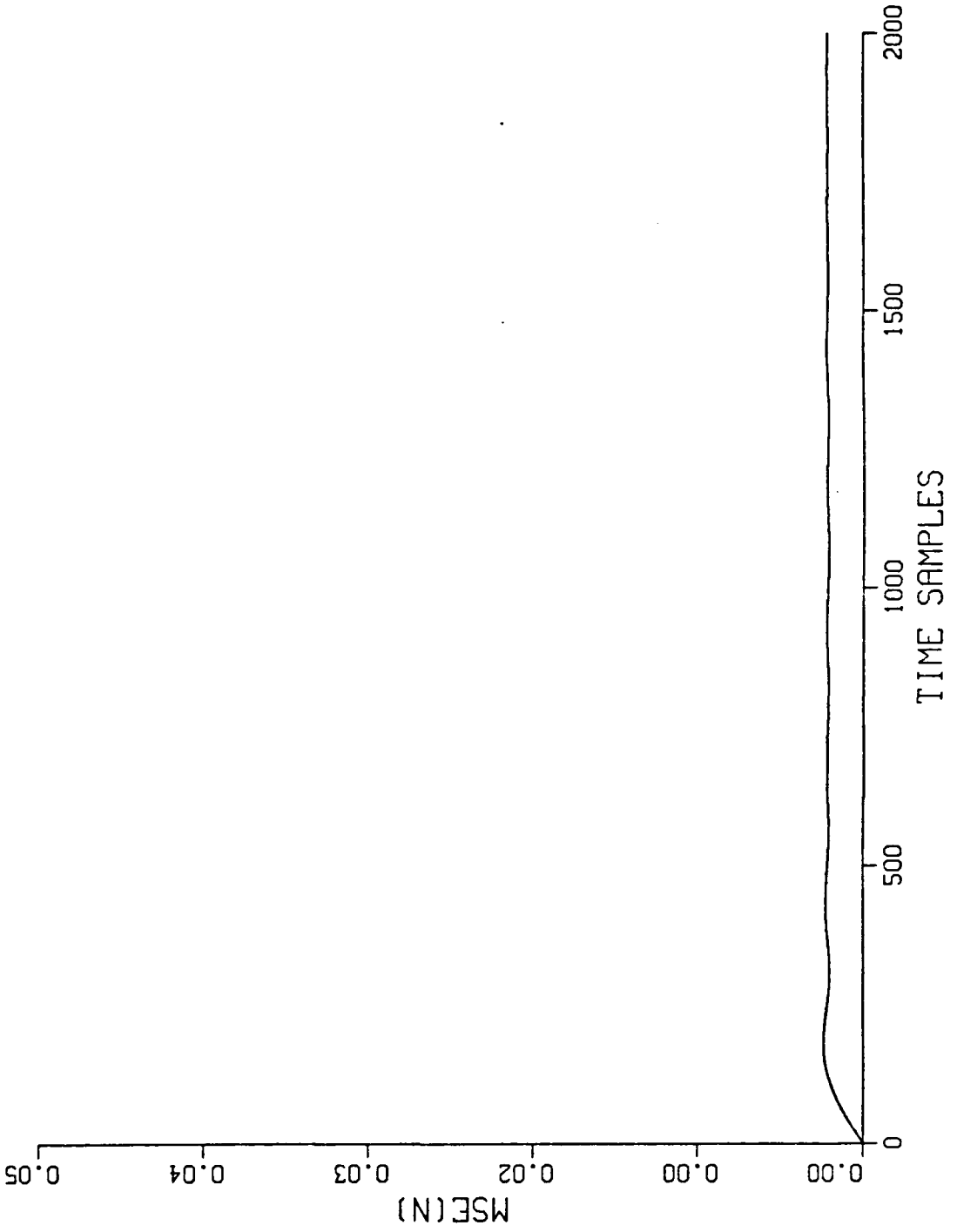


Figure 2.14 Mean-Square Error Between x(n) and $\hat{x}(n)$

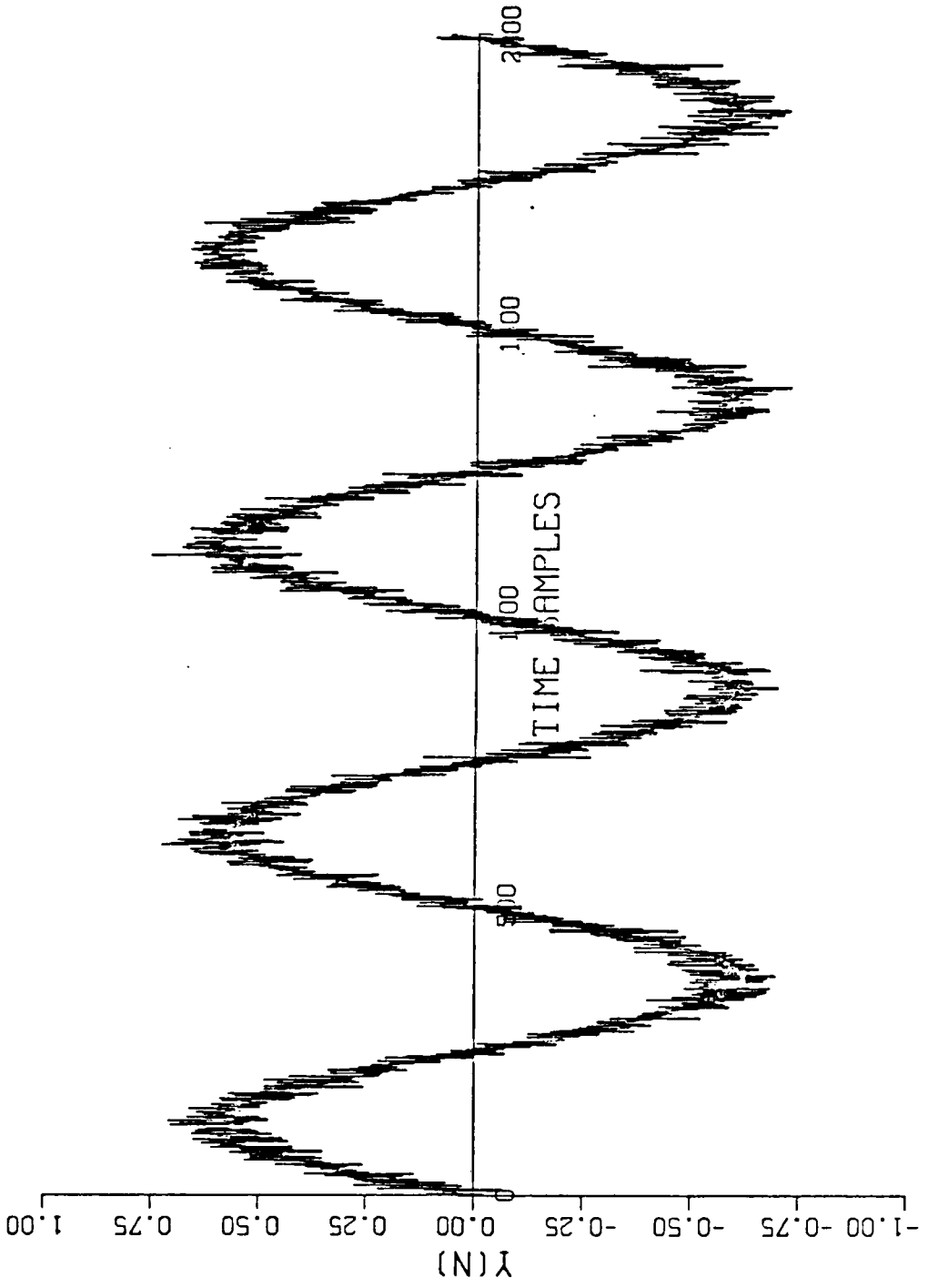


Figure 2.15 System Output Plus Measurement Noise

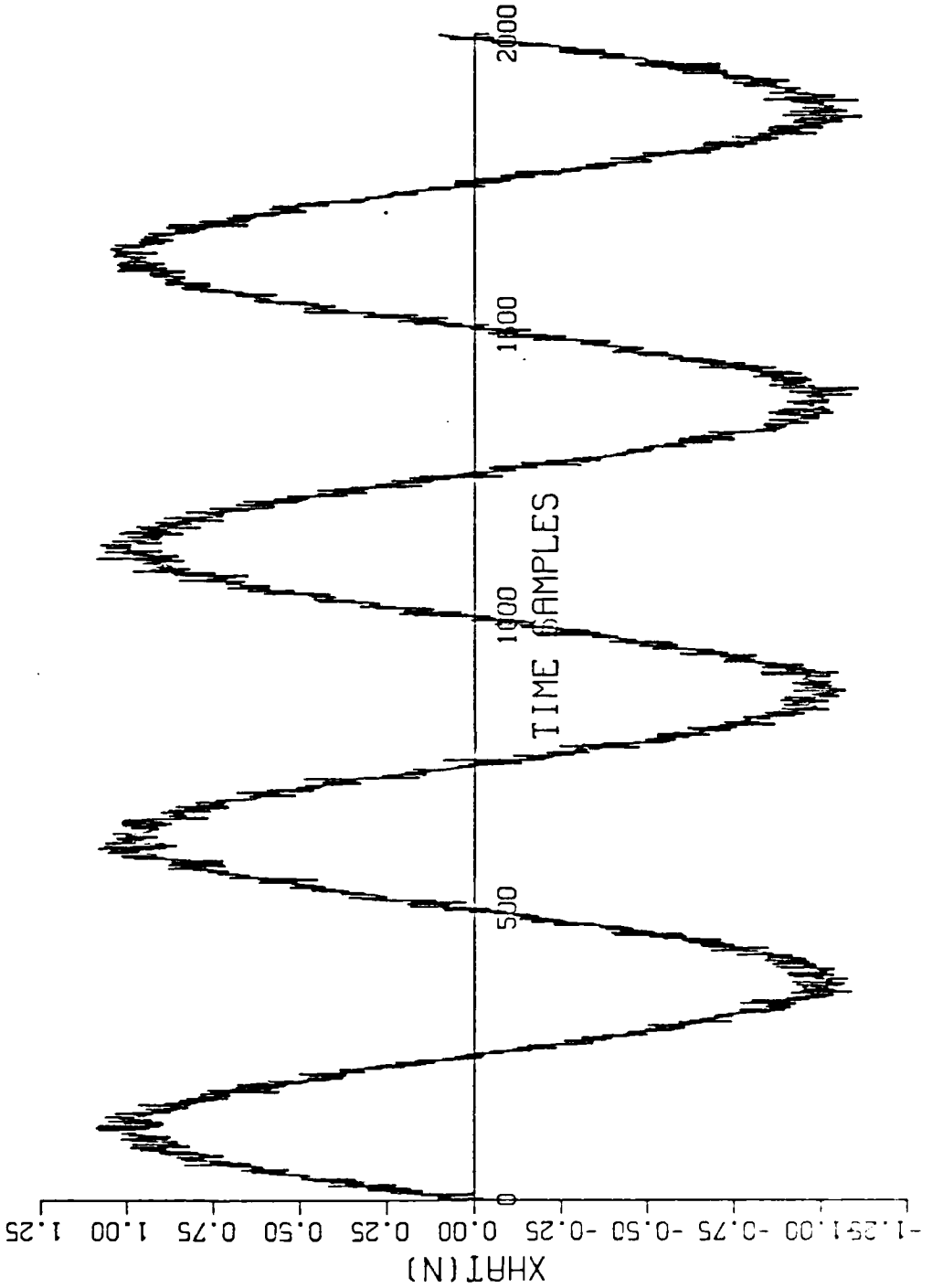


Figure 2.16 Lattice Filter Output, $\hat{x}(n)$

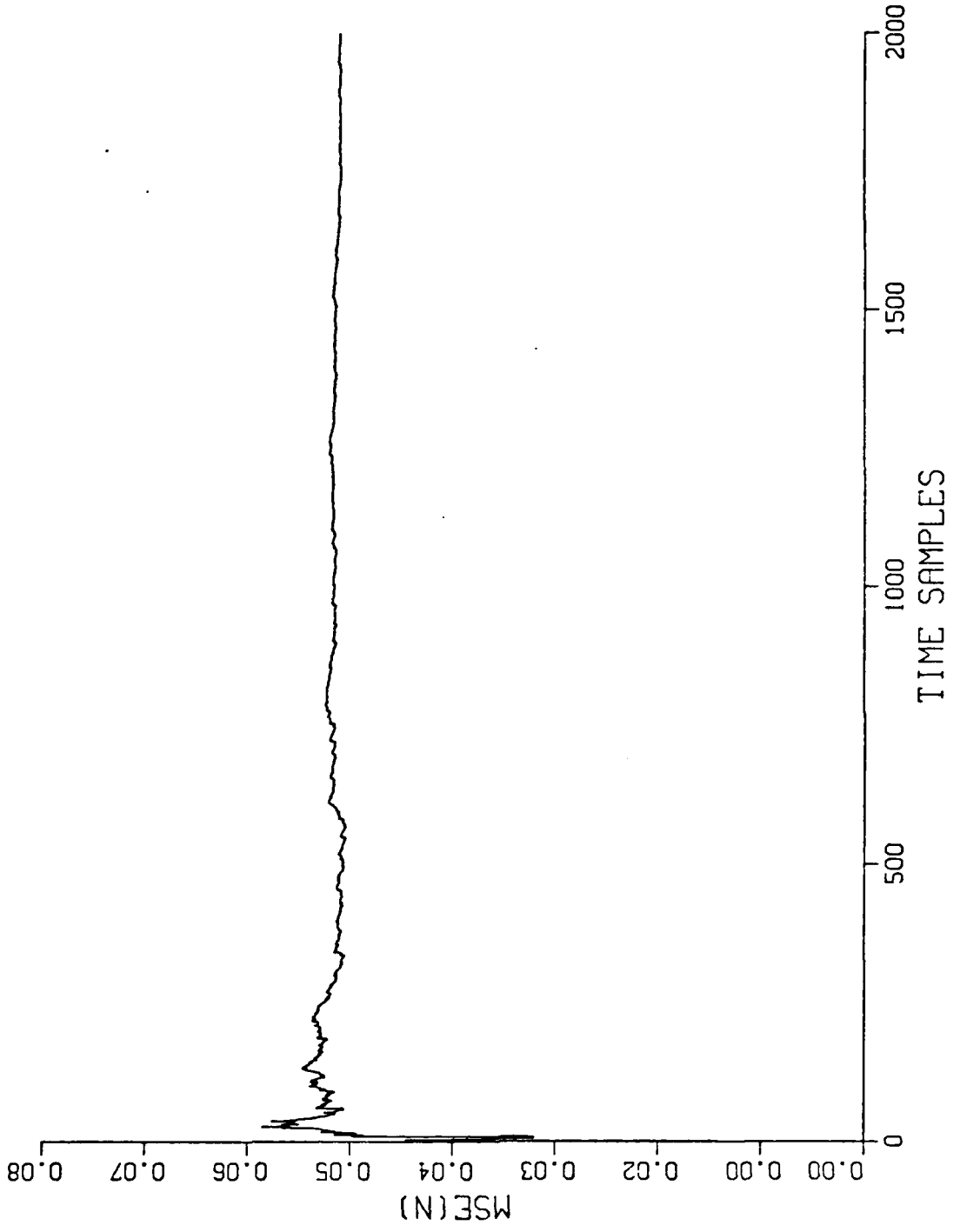


Figure 2.17 Mean-Square Error Between x(n) and $\hat{x}(n)$

III. NONLINEAR DECONVOLUTION

A. INTRODUCTION TO NONLINEAR SYSTEMS

While the previous chapter dealt solely with linear systems or plants, this chapter will address the inverse filtering problem involving nonlinear systems. The chapter starts with a brief introduction to modeling nonlinear systems. Then it quickly proceeds to extend the generalized linear lattice filter results of section II.D.4 to a nonlinear analysis lattice filter. The nonlinear lattice filter is discussed in detail. To conclude, results from numerous simulations involving the lattice in deconvolution applications are presented.

System linearity is defined in terms of the principle of superposition. If the rule by which the system transforms the input $x(k)$ into the output $y(k)$ is represented by the operator T, then the system is said to be linear if and only if

$$T[ax_1(k) + bx_2(k)] = aT[x_1(k)] + bT[x_2(k)] \qquad (3.1)$$
$$= ay_1(k) + by_2(k)$$

for arbitrary constants a and b. The system is nonlinear if equation (3.1) is not satisfied. Estimation of the parameters of a nonlinear system is a complex problem.

One approach to the parameter estimation problem for nonlinear filters is the Bayesian approach. This approach leads to various approximate solutions for the parameter estimates. In the Bayesian approach, the parameters are considered to be random variables. The parameter vector $\underline{h}$ is considered a random vector with a probability density function $p(\underline{h})$. Introducing the observation vector $\underline{y}(t)$ and the input vector $\underline{x}(t)$, both of which contain data up to time t, the a posteriori probability density function for $\underline{h}$ is $p(\underline{h}|\underline{y}(t),\underline{x}(t))$. One possible choice for the estimate of the $\underline{h}$ vector is to select the conditional mean $\hat{\underline{h}}(t) = E[\underline{h}|\underline{y}(t),\underline{x}(t)]$. Selecting this as the estimate minimizes the variance of the parameter estimation error. Another possible choice is to select the $\hat{\underline{h}}(t)$ which maximizes $p(\underline{h}|\underline{y}(t),\underline{x}(t))$. This most likely value is known as the maximum a posteriori (MAP) estimate. Finally, the Bayesian approach also leads to the extended Kalman filter for nonlinear state estimation problems. [Ref. 13:pp. 32-41]

Another popular method for modeling nonlinear systems is based on the Volterra series. The series is named after the mathematician Vito VOLTERRA. The first person to apply the series to nonlinear systems was Norbert WIENER. If the "black box" approach is taken towards a nonlinear, time-invariant system, the relationship between the input x(t) and the output y(t) can be represented by the Volterra

series:

$$
\begin{aligned}
y(t) = & \int_{-\infty}^{\infty} h_1(T_1)\,x(t-T_1)\,dT_1 \\
& + \iint_{-\infty}^{\infty} h_2(T_1,T_2)\,x(t-T_1)\,x(t-T_2)\,dT_1\,dT_2 \\
& + \iiint_{-\infty}^{\infty} h_3(T_1,T_2,T_3)\,x(t-T_1)\,x(t-T_2)\,x(t-T_3)\,dT_1\,dT_2\,dT_3 \\
& + \quad \ldots \\
& + \int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} h_n(T_1,\ldots,T_n)\,x(t-T_1)\ldots x(t,T_n)\,dT_1\ldots dT_n \\
& + \quad \ldots
\end{aligned}
\tag{3.2}
$$

where $n = 1,2,\ldots$ and $h_n(T_1,\ldots,T_n) = 0$ for any $T_j < 0$, $j = 1,2,\ldots,n$. The functions $h_n(T_1,\ldots,T_n)$ are called Volterra kernels of the system. This equation is a functional series. That is, it performs an operation on the function $x(t)$ which results in a number for $y(t)$. If the n-th order Volterra operator, H_n, is introduced where

$$
\begin{aligned}
y_n(t) &= H_n[x(t)] \\
&= \int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} h_n(T_1,\ldots,T_n)\,x(t-T_1)\ldots x(t-T_n)\,dT_1\ldots dT_n ,
\end{aligned}
\tag{3.3}
$$

then the series can be expressed in operator notation as

$$
\begin{aligned}
y(t) &= H_1[x(t)] + H_2[x(t)] + \ldots + H_n[x(t)] + \ldots \\
&= \sum_{n=1}^{\infty} H_n[x(t)] = \sum_{n=1}^{\infty} y_n(t) .
\end{aligned}
\tag{3.4}
$$

Figure 3.1 is a graphic representaion of this equation. [Ref. 14:pp. 7-9] The H_1 operator is a linear operator, H_2 is a quadratic operator; and, in general, the H_n operator involves the term x(t) to the n-th power.

B. GENERALIZED NONLINEAR LATTICE FILTER

1. Introduction

In this section, it will be demonstrated how the generalized lattice filter introduced in section II.D.4 can be applied to modeling nonlinear systems. The development will start by looking at the discrete form of the Volterra series. Based upon this series an alternate tensor notation representation will be introduced. The results of section II.D.4 will then be extended to handle a two-dimensional field of data which will result in the generalized nonlinear lattice filter.

2. Nonlinear Lattice Filter Development

The discrete form of the Volterra series of equation (3.2) is given by

$$y(k) = h_0 + \sum_{n_1} h_1(n_1)x(k-n_1) + \sum_{n_1}\sum_{n_2} h_2(n_1,n_2)x(k-n_1)x(k-n_2) + \ldots \quad (3.5)$$

Using LENK's tensor notation, an equivalent form of equation (3.5) is given by

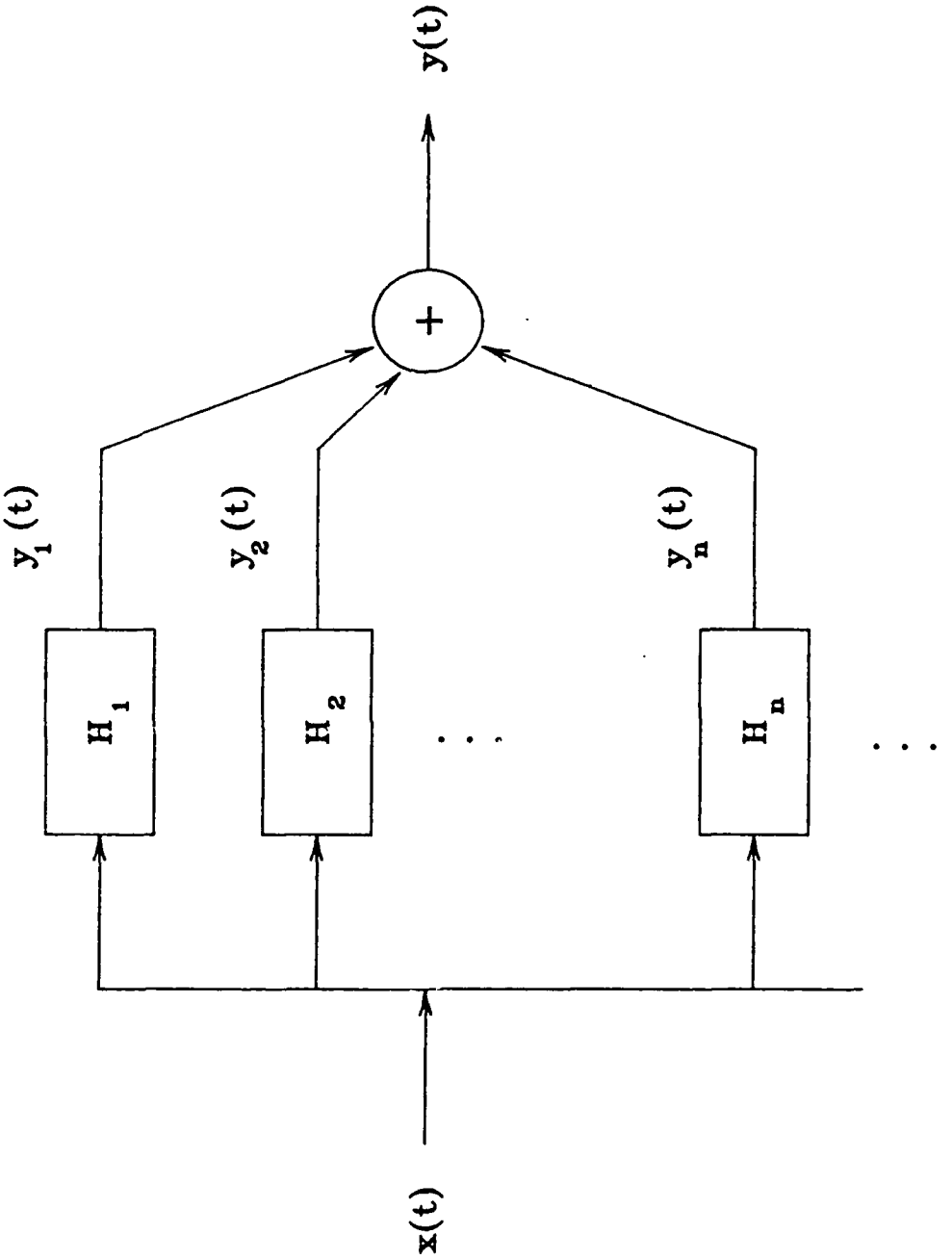


Figure 3.1 Volterra Series Representation

$$y(k) = H + H_{\lambda_1} \underline{x}^{\lambda_1} + H_{\lambda_1\lambda_2} \underline{x}^{\lambda_1} \underline{x}^{\lambda_2} + \ldots \text{, where} \qquad (3.6a)$$

$$\underline{x}^{\lambda_i}(k) = [x(k), x(k-1), \ldots, x(k-\lambda_i), \ldots, x(k-N)]^T \qquad (3.6b)$$

for $\lambda_i = 0,1,\ldots,N$. As an example, if N=1 and equation (3.6a) is truncated to the three terms shown above, then this equation can be rewritten as

$$\begin{aligned} y(k) = {} & H_0 + H_1 x(k) + H_1 x(k-1) + H_{00} x(k)x(k) \\ & + H_{01} x(k)x(k-1) + H_{10} x(k-1)x(k) \\ & + H_{11} x(k-1)x(k-1) . \end{aligned} \qquad (3.7)$$

Based on this tensor notation, LENK introduced an alternate representation for the nonlinear system. Instead of defining the components of the vector $\underline{x}(k)$ as the present and past values of the signal x(k) as in equation (3.6b), the components are redefined in terms of x(k) raised to the λ_i power for $\lambda_i = 0,1,2,\ldots,N$. That is

$$\begin{aligned} \underline{x}^{\lambda_i}(k) &= [x^0(k), x^1(k), x^2(k), \ldots, x^N(k)]^T \qquad (3.8) \\ &= [1, x(k), x^2(k), \ldots, x^N(k)]^T \end{aligned}$$

for $\lambda_i = 0,1,2,\ldots,N$. Now the output of the nonlinear system is defined by

$$y(k) = x^{\lambda_0}(k) \ldots x^{\lambda_N}(k-N) H_{\lambda_0 \ldots \lambda_N} \qquad (3.9)$$

for $\lambda_0, \lambda_1, \ldots = 0,1,2,\ldots,P$. The variable P gives the order of the filter and N defines the filter's finite memory. The term $H_{\lambda_0 \cdots \lambda_N}$ plays a role similar to that of the Volterra kernel; it can be considered a (P+1)-order tensor. To clarify the meaning of this notation, the following example with N=1 and P=2 is provided:

$$
\begin{aligned}
y(k) &= H_{\lambda_0 \lambda_1} x^{\lambda_0}(k) x^{\lambda_1}(k-1) \\
&= H_{00} + H_{10} x(k) + H_{01} x(k-1) + H_{11} x(k)x(k-1) + H_{20} x^2(k) \\
&\quad + H_{02} x^2(k-1) + H_{21} x^2(k)x(k-1) + H_{12} x(k)x^2(k-1) \\
&\quad + H_{22} x^2(k) x^2(k-1) \, . \qquad (3.10)
\end{aligned}
$$

[Ref. 12:pp. 39-48]

The above nonlinear model for y(k) is a moving average (MA) model: the output is defined in terms of the input signal. The next step is to establish an autoregressive (AR) model where y(k) is estimated in terms of its past values. This type of model is useful when the input signal is not readily available. An autoregressive model is obtained by redefining the observation vector in terms of y(k) vice x(k). Then the AR model is given by

$$
y(k) = y^{\lambda_1}(k-1) \ldots y^{\lambda_N}(k-N) H_{\lambda_1 \cdots \lambda_N} \qquad (3.11)
$$

for $\lambda_i = 1,2,\ldots,P$. If the system is driven by a white noise signal u(k), and if the system's parameters are

approximated by $\hat{H}_{\lambda_0 \cdots \lambda_N}$, then the estimation error is

$$e(k)=y(k)-\hat{y}(k)=[y^{\lambda_1}(k-1)\ldots y^{\lambda_N}(k-N)H_{\lambda_1 \cdots \lambda_N} + u(k)]$$

$$- [\, y^{\lambda_1}(k-1)\ldots y^{\lambda_N}(k-N)\hat{H}_{\lambda_1 \cdots \lambda_N}] \quad (3.12)$$

If the system parameters are known exactly, then the estimation error and the white noise sequences are equal--that is $e(k)=u(k)$. One note concerning the nonlinear AR model: unlike the linear AR model whose stability is easily determined (i.e., the system is stable if all its poles lie within the unit circle in the complex z-plane), it is difficult to judge for which class of inputs the AR nonlinear system's output will remain bounded. This is because the order of the nonlinearity increases with time. [Ref. 12:pp. 68-69]

Using this alternate tensor form of the AR nonlinear system, along with the generalized lattice of section II.D.4, the nonlinear lattice filter will be developed. To simplify the discussion, the filter's finite memory will be restricted to N=2. Then the product $\underline{Y}(k) = [y^{\lambda_1}(k-1)y^{\lambda_2}(k-2)]$ for $\lambda_1, \lambda_2 = 0,1,\ldots,P$ forms a second order tensor, or a two-dimensional data field given by:

$$\underline{Y}(k) = \begin{vmatrix} 1 & y(k-2) & \cdots & y^{P}(k-2) \\ y(k-1) & y(k-1)y(k-2) & \cdots & y(k-1)y^{P}(k-2) \\ y^{2}(k-1) & y^{2}(k-1)y(k-2) & \cdots & y^{2}(k-1)y^{P}(k-2) \\ \vdots & \vdots & & \vdots \\ y^{P}(k-1) & y^{P}(k-1)y(k-2) & \cdots & y^{P}(k-1)y^{P}(k-2) \end{vmatrix} \quad (3.13)$$

The types of systems that this nonlinear lattice structure can model exactly are of the form shown in Figure 3.2 . The nonlinear combinations block forms a weighted sum of the cross-products and powers of the input variables y(k-i), for i=1,2,...,N. As an example, with N=2 and P=2 the general equation for the system's output when excited by the input x(k) is given by:

$$\begin{aligned} y(k) = x(k) - \{ & H_{11} + H_{21} y(k-1) + H_{12} y(k-2) \\ & + H_{22} y(k-1)y(k-2) + H_{31} y^{2}(k-1) + H_{32} y^{2}(k-1)y(k-2) \\ & + H_{13} y^{2}(k-2) + H_{23} y(k-1)y^{2}(k-2) \\ & + H_{33} y^{2}(k-1)y^{2}(k-2) \} \end{aligned} \quad (3.14)$$

It is also assumed that the system is time-invariant; otherwise the model changes with time k. In order to define the forward and backward prediction errors, the $(P+1)^2$ elements of the two-dimensional data field must by converted

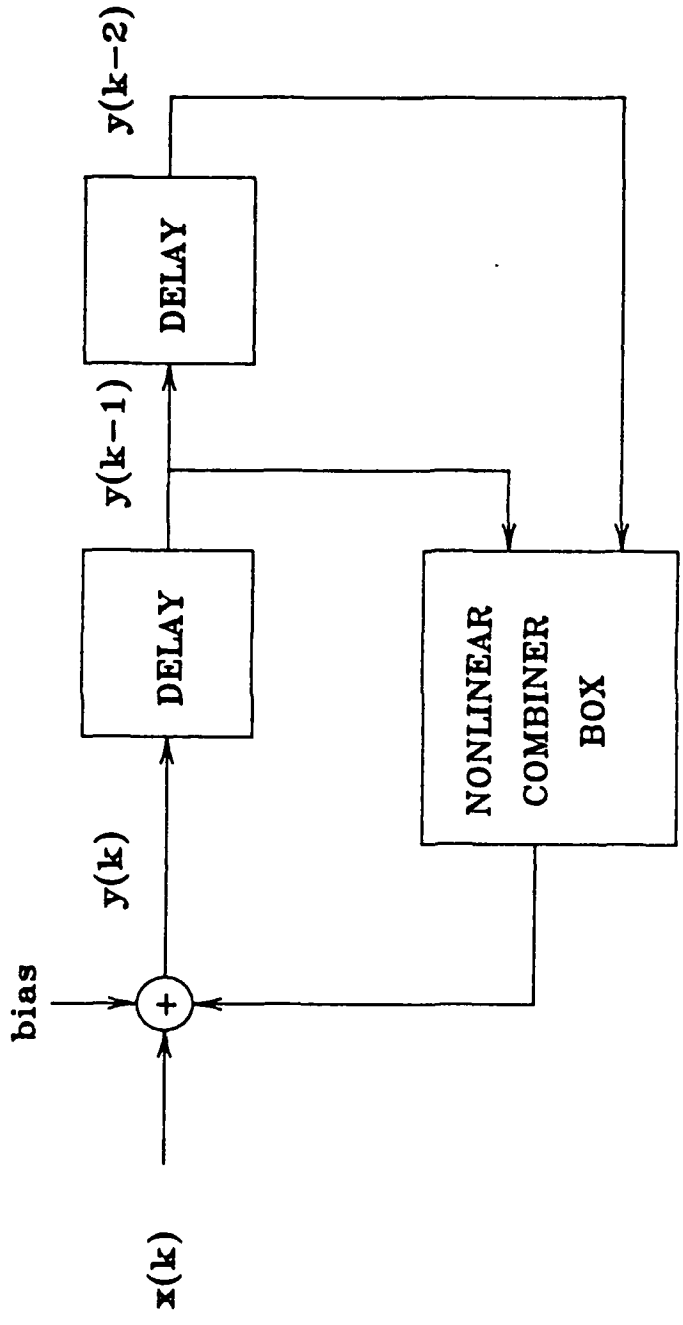


Figure 3.2 Nonlinear System Model

into a one-dimensional sequence. The ordering chosen by LENK is listed below:

$$\begin{aligned} \text{order} = \{&(P,P),(P-1,P),(P,P-1),\ldots,(0,P),(P,0),\\ &(P-1,P-1),(P-2,P-1),(P-1,P-2),\ldots,(0,P-1),\\ &(P-1,0),\ldots,(1,1),(0,1),(1,0),(0,0)\} \quad , \end{aligned} \tag{3.15}$$

where the components of the data field are identified by the indices (m,n) for $m,n = 0,1,\ldots,P$. The elements of the ordered set are numbered consecutively from 0 to $(P+1)^2-1$. The notation $(m,n)-q$ is used to identify the q-th element prior to the element (m,n) as referenced to the ordered sequence. This notation will be further abbreviated to $mn-q$. Now the $(q-1)$-order, normalized, forward error associated with predicting the value of the element $y_{m\,n}(k-1)y(k-2)$ from the preceding $(q-1)$ elements is given by

$$\bar{e}^{\,q-1}_{mn} = a^{\,q-1}_{\lambda_1\lambda_2}(m,n)\,y^{\lambda_1}(k-1)\,y^{\lambda_2}(k-2) \tag{3.16}$$

for $\lambda_1, \lambda_2 = 0,1,2,\ldots,P$. Note that the coefficients $a^{\,q-1}_{\lambda_1\lambda_2}$ can be thought of as the components of a second order tensor. The coefficient $a^{\,q-1}_{\lambda_1\lambda_2}$ is equal to zero when the indices (λ_1,λ_2) do not correspond to the $(q-1)$ elements preceding (m,n) in the ordered sequence (i.e., when $(\lambda_1,\lambda_2) > (m,n)$ or when $(\lambda_1,\lambda_2) \leq mn-q$). Also, when $(\lambda_1,\lambda_2) = (m,n)$, then

$$a^{\,q-1}_{mn}(m,n) = 1 \,/\, \| e^{\,q-1}_{mn} \| \quad . \tag{3.17}$$

Similarly, the normalized backward prediction error in estimating y(mn-q) from the next (q-1) points is given by

$$\bar{r}^{q-1}_{mn-q} = b^{q-1}_{\lambda_1\lambda_2}(mn-q)\,y^{\lambda_1}(k-1)\,y^{\lambda_2}(k-2) \tag{3.18}$$

for $\lambda_1,\lambda_2 = 0,1,2,\ldots,P$. The coefficients $b^{q-1}_{\lambda_1\lambda_2}(mn-q)$ equal zero when the indices (λ_1,λ_2) do not correspond to the (q-1) elements following (m,n) (that is, when $(\lambda_1,\lambda_2) < (mn-q)$ or $(\lambda_1,\lambda_2) > (m,n)$). When $(\lambda_1,\lambda_2) = (m,n)$, then

$$b^{q-1}_{mn-q}(mn-q) = 1/\,\|r^{q-1}_{mn-q}\|. \tag{3.19}$$

[Ref. 12:pp. 145-146]

Using the normalized nonlinear Levinson algorithm, LENK shows that the nonlinear prediction errors can be updated in order through the recursion relation

$$\begin{vmatrix} \bar{e}^{q}_{mn} \\ \bar{r}^{q}_{mn-q} \end{vmatrix} = \theta(K^{mn}_{q}) \begin{vmatrix} \bar{e}^{q-1}_{mn} \\ \bar{r}^{q-1}_{mn-q} \end{vmatrix} \tag{3.20}$$

where

$$\theta(K^{mn}_{q}) = \frac{1}{\sqrt{1-(K^{mn}_{q})^2}} \begin{vmatrix} 1 & -K^{mn}_{q} \\ -K^{mn}_{q} & 1 \end{vmatrix} \tag{3.21}$$

and the unique reflection coefficients are given by

$$K_{q}^{mn} = E\{\bar{e}_{mn}^{q-1}\,\bar{r}_{mn-q}^{q-1}\} \quad . \tag{3.22}$$

[Ref. 12:pp. 146-147]

A deficiency remains to be corrected: the goal is to estimate y(k), but there is no y(k) term in the two-dimensional data matrix of equation (3.13). This problem is solved by adding another channel to the lattice structure and by exploiting the orthogonality of the backward prediction errors. Since the backward prediction errors leaving the top row of the lattice filter (Figure 2.9) are uncorrelated, they can be used in a Fourier series to estimate y(k). These backward prediction errors can be formed into a length $L = (P+1)^2$ vector defined as

$$[\bar{r}_{(m,n)-\lambda'}^{\lambda'}] = [\bar{r}_{00}^{0}, \bar{r}_{00-1}^{1}, \bar{r}_{00-2}^{2}, \ldots, \bar{r}_{00-L+1}^{L-1}]^T \tag{3.23}$$

where the subscript is in the form mn-q. Now, the error in estimating y(k) using the data in the $\underline{Y}(k)$ tensor is calculated from

$$e_k^L = y(k) - \sum_{\lambda'=0}^{L-1} K_{\lambda'} y^{\lambda'}, \tag{3.24}$$

where the Fourier coefficients, $K_{\lambda'}$, are given by

$$[K_{\lambda}]=[E\{y(k)\bar{r}_{00}^{0}\},E\{y(k)\bar{r}_{00-1}^{1}\},\ldots,E\{y(k)\bar{r}_{00-L+1}^{L-1}\}]. \quad (3.25)$$

The resulting generalized nonlinear analysis lattice structure is depicted in Figure 3.3. [Ref. 12:pp. 147-150]

The nonlinear lattice structure will now be examined more closely. As can be seen in Figure 3.3 , the inputs to the analysis filter are the normalized values of y(k) and the $(P+1)^2+1$ ordered components of the two-dimensional data field (equation (3.13)). The ordering of these inputs is in accordance with equation (3.15). At a given time k, these $(P+1)^2+2$ inputs are evaluated and inserted into the left side of the lattice structure. The lattice calculations are conducted by starting at the top row and moving downward along the northwest-southeast diagonal, and then advancing to the top of the next diagonal and so on. When all the lattice calculations for time k have been completed, the process is repeated for time k+1. Just as in the case of the linear lattice filter, the PARCOR coefficients at each lattice section act to decorrelate the two input signals. The backward error signals exit at the top of the lattice structure, and the forward error signals exit at the right side. The output of interest for inverse filtering is the forward error signal from the top row of the filter (i.e., the row corresponding to the y(k) input signal). This is the error arising from the estimation of y(k).

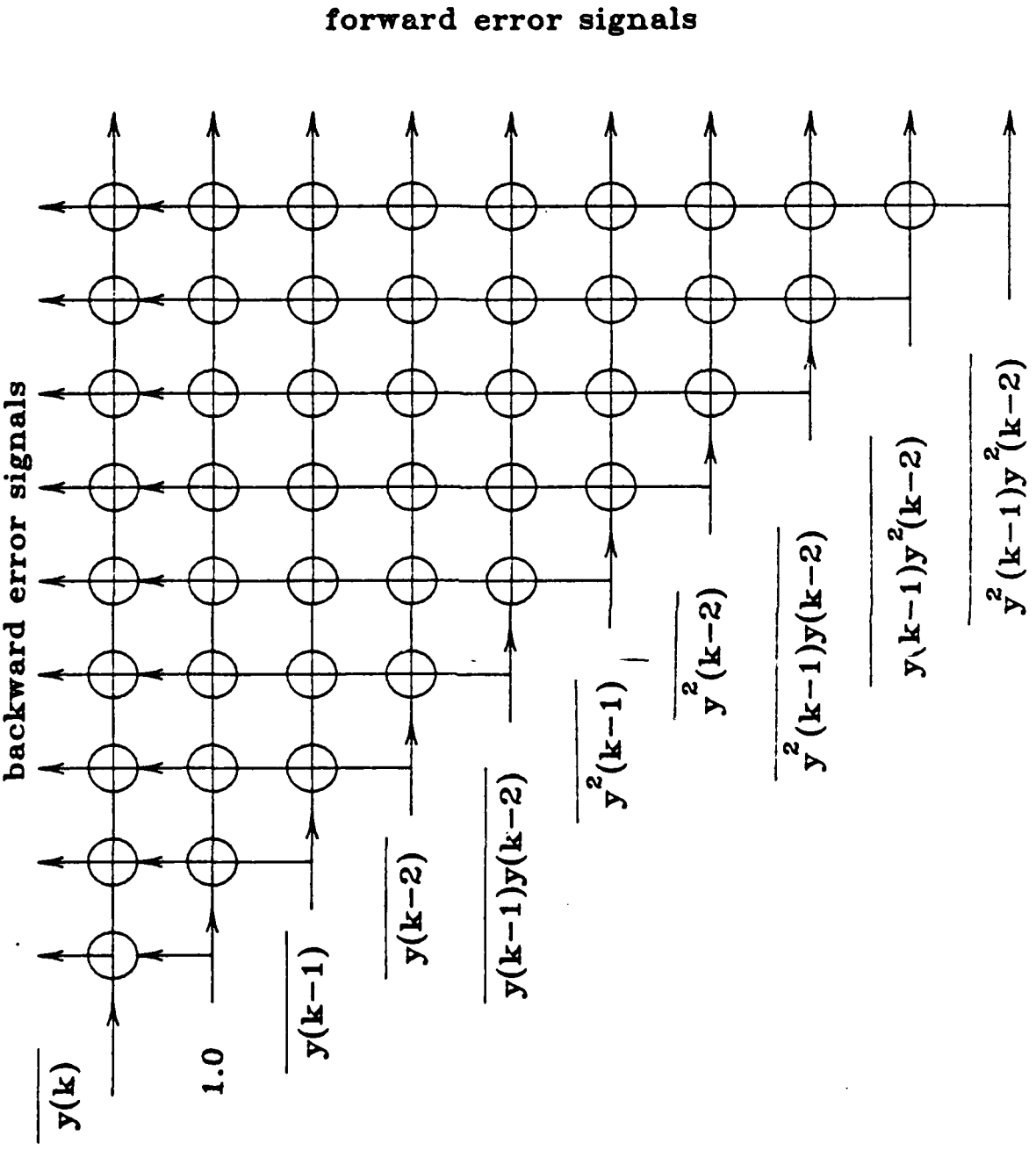


Figure 3.3 Quadratic Nonlinear Lattice Filter

3. The Nonlinear Lattice Applied to Deconvolution

The generalized nonlinear lattice filter can be applied to the inverse filtering problem just as the linear lattice filter was in section II.D.6. The first problem is one of system identification and parameter estimation. Once the model parameters have been estimated, they are embedded in the nonlinear filter lattice structure. Then, the nonlinear lattice represents the inverse of the system's transfer function. As will be shown, the nonlinear lattice filter can model both linear and nonlinear systems-- the linear system is just a special case of the nonlinear system. This inverse filtering algorithm was implemented by the FORTRAN programs NLMAIN, NLCLAT, SCHUR, NORMS, NLLAT, and URAND. These programs are listed in Appendix B. Now, this inverse filtering procedure will be described in more detail.

In order to identify the model parameters (i.e., the PARCOR coefficients), the system was excited with zero mean, unit variance white noise sequences. Both Guassian and uniform noise distributions were used with good results. One difficulty encountered in generating the nonlinear data was ensuring that the output of the postulated nonlinear system remained bounded for the input noise signal. For the simulations, the filter and system were constrained to a finite memory of at most two delays (N=2), while the

nonlinear order was allowed to vary from P=0 to P=4. (For N greater than two, the problem of ordering the elements of the N-dimensional $\underline{Y}$ tensor increases in complexity.) The system output sequence y(k) was then processed by subroutines NLCLAT and SCHUR, which determined the corresponding autocorrelation matrix and the partial correlation coefficients, respectively. In an effort to improve the accuracy of these calculations, noise sequences of up to 5,000 points were used. Additionally, the PARCOR coefficients were averaged over as many as 50 realizations of the input noise random process. Through trial and error, it was found that the best inverse filtering results were obtained when the resulting reflection coefficients were truncated to two decimal places. The output of subroutine SCHUR is a $((P+1)^2+1 \times (P+1)^2+1)$ upper triangular matrix of reflection coefficients. This matrix is simply overlaid atop the upper triangular shaped nonlinear lattice structure to place the reflection coefficients at the correct filter sections.

With the reflection coefficients calculated and embedded in the filter, we were able to use the lattice in an inverse filtering application. To recover the input signal x(k), the system's output signal y(k) was processed by subroutines NORMS and NLLAT. The function of NORMS was to normalize the $(P+1)^2+1$ input signals into the lattice.

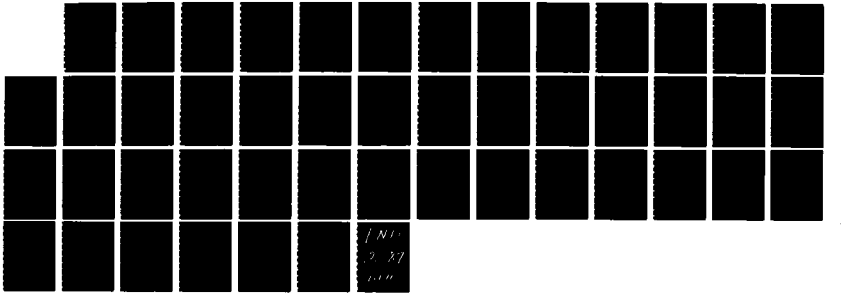
Then subroutine NLLAT, using the previously calculated reflection coefficients, implemented the lattice structure and carried out the lattice filter calculations. The forward error signal out of the top row of the lattice yielded the normalized estimate of x(k). Since the inputs to the lattice filter are all normalized, the outputs must be denormalized. Therefore, since the input into the top row of the lattice is divided by the norm of y(k), the forward error signal from the top row of the lattice is multiplied by the norm of y(k). (Note that if y(k) is a zero mean sequence, then this norm is equivalent to its standard deviation.) Also, it was found that to achieve a good estimate, it was necessary to further scale this error signal by dividing it by the norm of the noise generated sequence y(k).

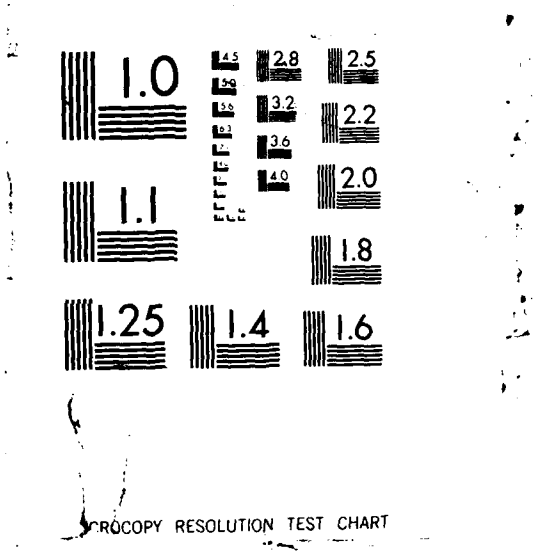
In order to verify the accuracy of the reflection coefficients, the noise generated output signal, y(k), was passed through the inverse lattice filter to see how well the filter whitened this sequence. The effectiveness of the whitening filter was evaluated by examining the mean-square error (MSE) between the white noise input signal, x(k), and the lattice filter's output, $\hat{x}(k)$. The running average mean-square error was calculated using the equation

$$MSE(k) = \sqrt{(1/k) \sum_{i=1}^{k} (x(i) - \hat{x}(i))^2} \quad . \qquad (3.26)$$

AD-A175 147 ORTHOGONAL LATTICE MODELING OF NONLINEAR SYSTEMS(U) 2/2
NAVAL POSTGRADUATE SCHOOL MONTEREY CA S L JOHNSON
SEP 86

UNCLASSIFIED F/G 9/3 NL




1.0
1.1
1.25
1.4
1.6
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.8
CROCOPY RESOLUTION TEST CHART

Since the noise input, x(k), has unit variance, the MSE plot essentially provides a percentage error between $\hat{x}(k)$ and x(k). Plots of the MSE are included in the simulation results. Having demonstrated that the lattice filter represented a good inverse of the system H(z), the system was then driven by a known input signal x(k). To recover an approximation to this signal, the corresponding system output was passed through the lattice filter, and the resulting lattice filter output was denormalized and rescaled as previously discussed. Simulation results for various systems are presented in the following section.

4. Inverse Filtering Simulation Results

In this section, the previous modeling and inverse filtering procedures are implemented and applied to various linear and nonlinear systems. Unless otherwise noted, the reflection coefficients for each of the systems were determined by using twenty-five realizations (5,000 points each) of the zero mean, unit variance white noise random process as the input excitation signal. As previously mentioned, the mean-square error between this white noise input and the lattice filter's output was plotted to evaluate the lattice filter's inverse filtering performance. After the system was modeled, it was driven by the signal x(k) which consisted of ramps, pulses, and sinusoids as shown in Figure 3.4.

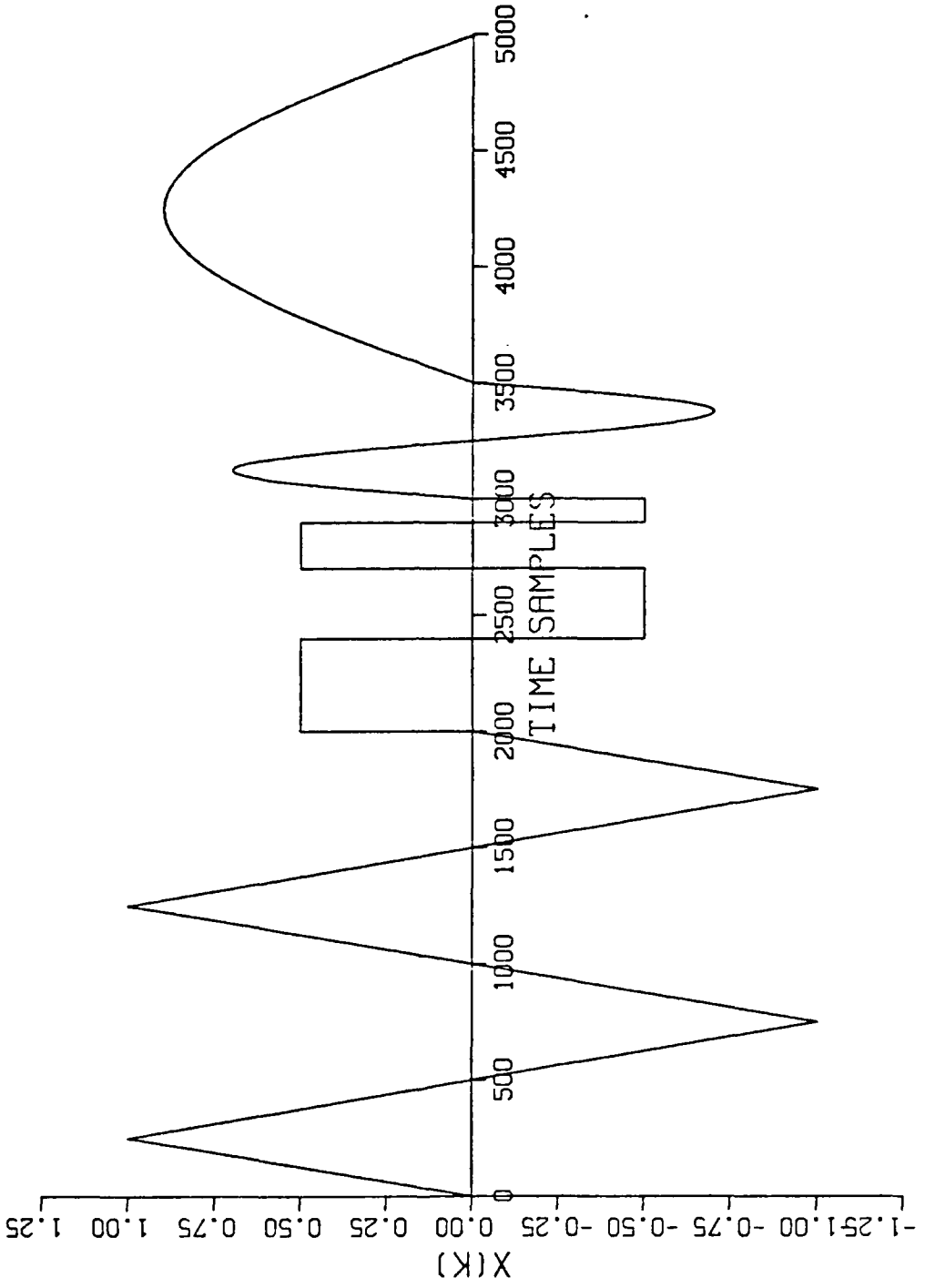


Figure 3.4 Input Signal x(k)

a. System I: $y(k) = x(k) - (0.6y(k-1) + 0.08y(k-2))$

This is the linear system first introduced in Section II.D.6. An input Gaussian white noise sequence was used to model the system. Figure 3.5 is a plot of the running average mean-square error between the input noise and output error signals. In Section II.D.6, the linear lattice filter's reflection coefficients were found to be $K_1 = -0.555$ and $K_2 = -0.077$. It should be noted that these values appear in the first row of the nonlinear lattice's reflection coefficient matrix. Here, for a first order, nonlinear lattice filter, the reflection coefficients are given by the upper triangular matrix

$$K = \begin{vmatrix} 0 & 0 & -.55 & -.07 & 0 \\ 0 & 0 & 0 & 0 & -.43 \\ 0 & 0 & 0 & -.55 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix} .$$

The system output, y(k), corresponding to the input signal of Figure 3.4 is shown in Figure 3.6. As can be seen by the plots of x(k) and $\hat{x}(k)$ in Figure 3.7, the nonlinear lattice filter did an outstanding job of recovering the "unknown" input signal x(k) from the linear system's output y(k).

b. System II: $y(k) = x(k) - (0.2y(k-1)y(k-2))$

This system involves a "cross-talk" nonlinearity, that is, the output is a function of the product of two different signals. For this system, both

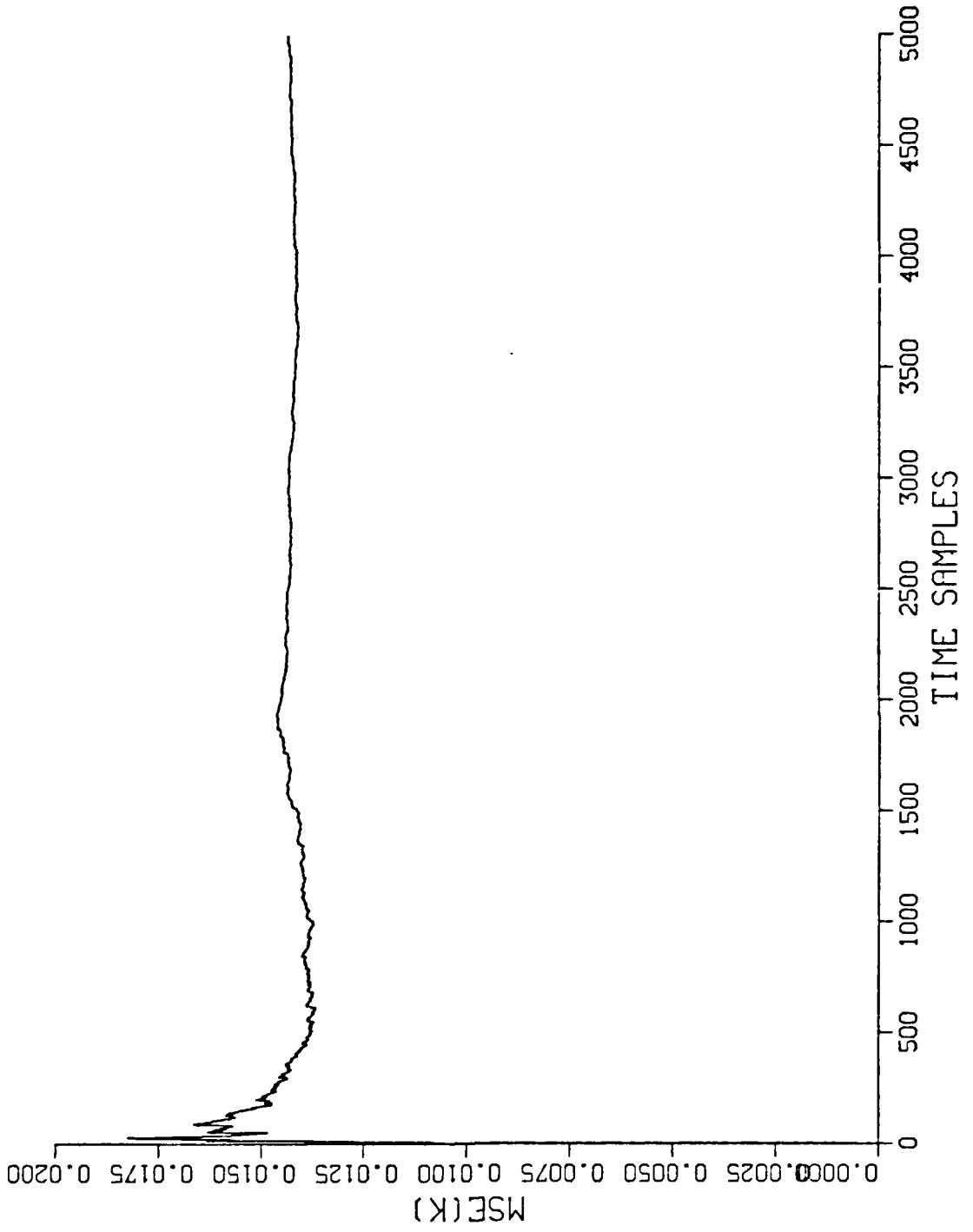


Figure 3.5 System I Mean-Square Error

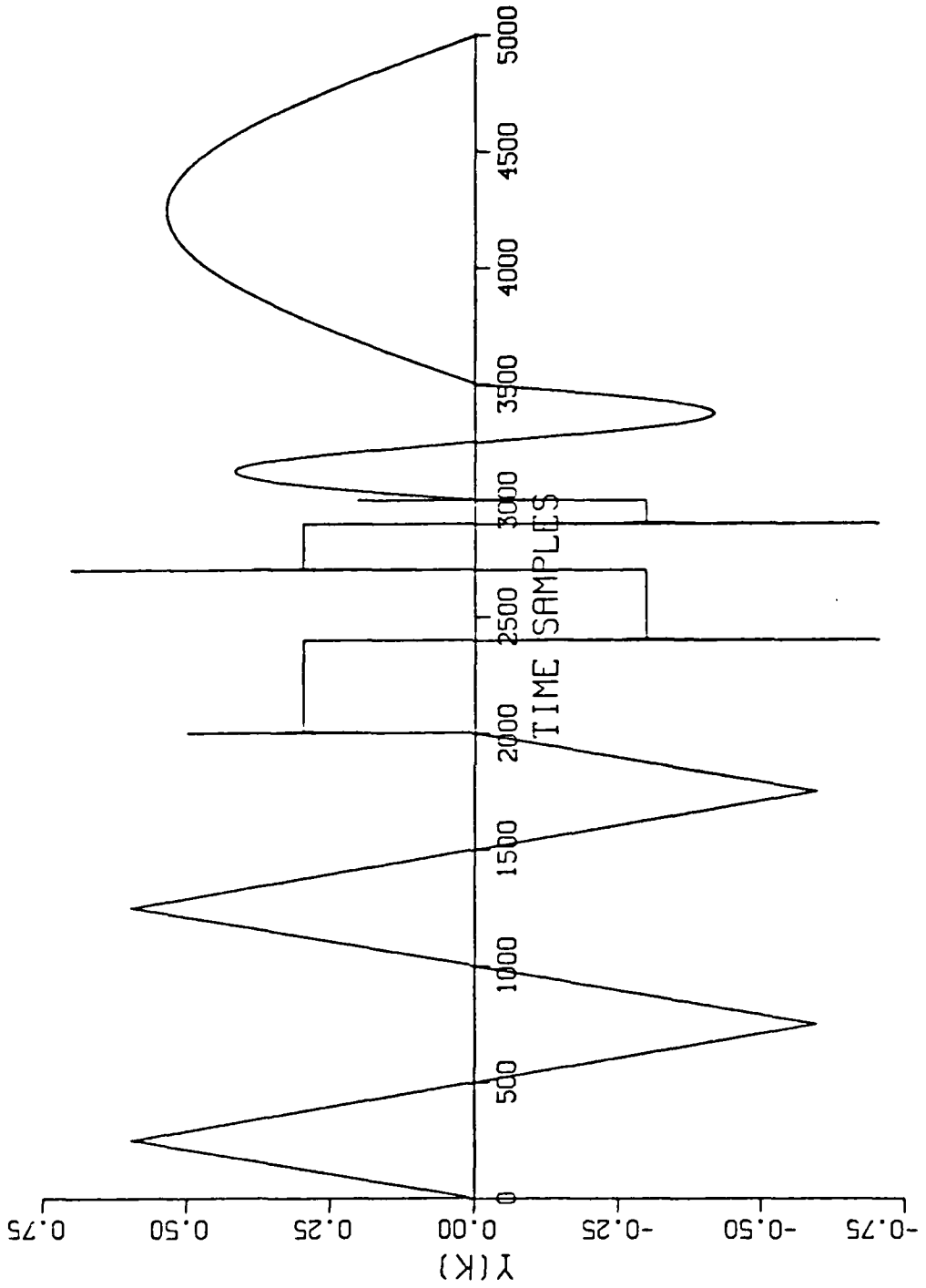


Figure 3.6 System I Output y(k)

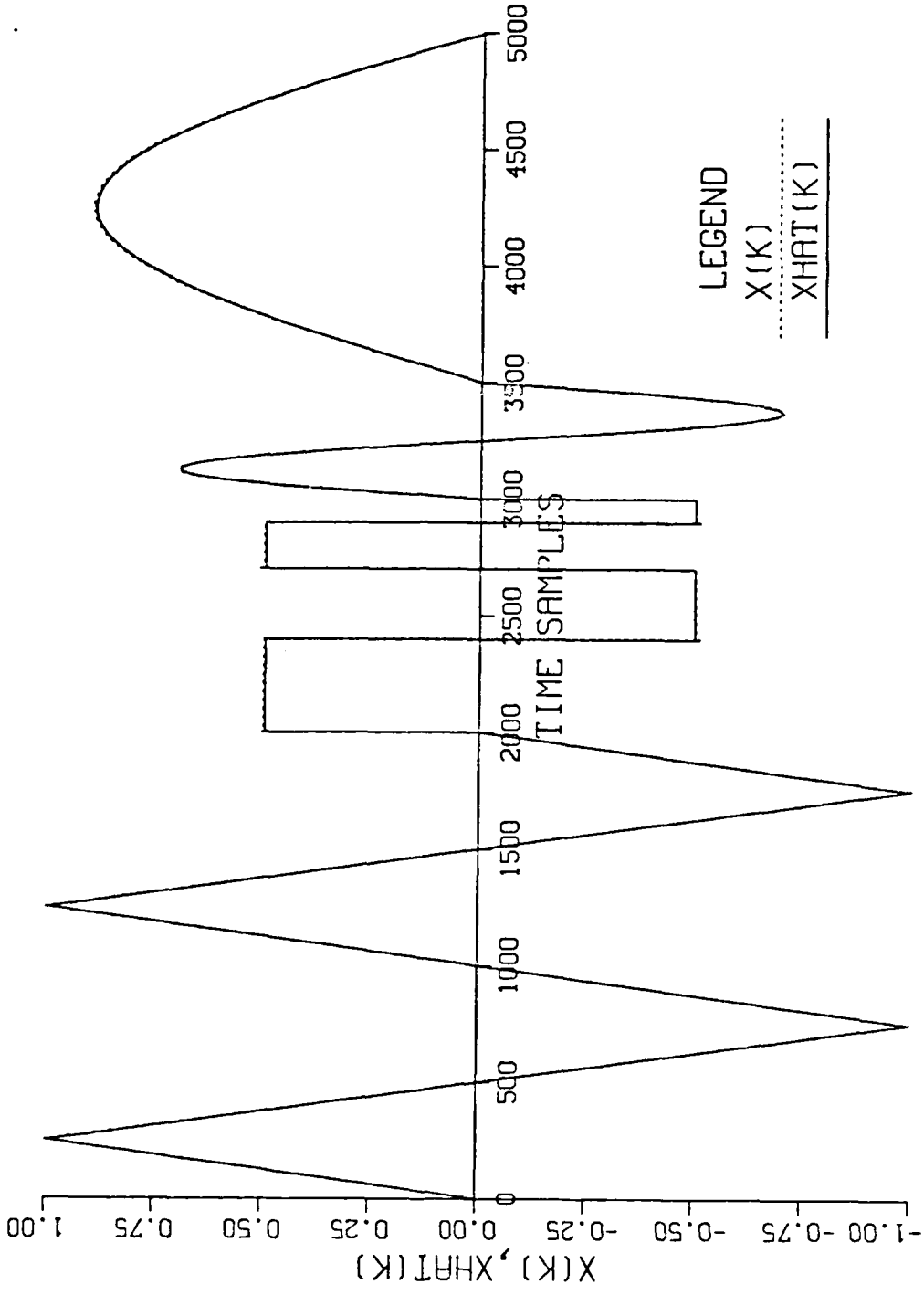


Figure 3.7 System I Comparison of x(k) and $\hat{x}(k)$

Gaussian and uniform noise distributions were used in the modeling process in order to compare the two techniques. Here, both methods yielded the same reflection coefficients:

$$K = \begin{vmatrix} 0 & 0 & 0 & 0 & -.21 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix} .$$

The only difference between the two techniques was the value of the scaling factor (i.e., the norm of the noise generated system output y(k)). The plot of y(k) is shown in Figure 3.8. The running average mean-square error and $\hat{x}(k)$ plots for the Gaussian noise derived model are depicted in Figures 3.9 and 3.10, respectively. The corresponding plots for the uniform noise derived model are given in Figures 3.11 and 3.12. As can be seen by comparing the plots, both modeling variations lead to nearly identical results. Both techniques yielded excellent approximations to the input signal x(k).

c. System III: $y(k) = x(k) - 0.2y(k-1)y(k-1)$

This system involves a quadratic nonlinearity. Therefore, a second order model was used. The system output would not remain bounded for a Gaussian noise input, so the system was modeled using a unit variance uniform noise random process. The resulting reflection coefficients are:

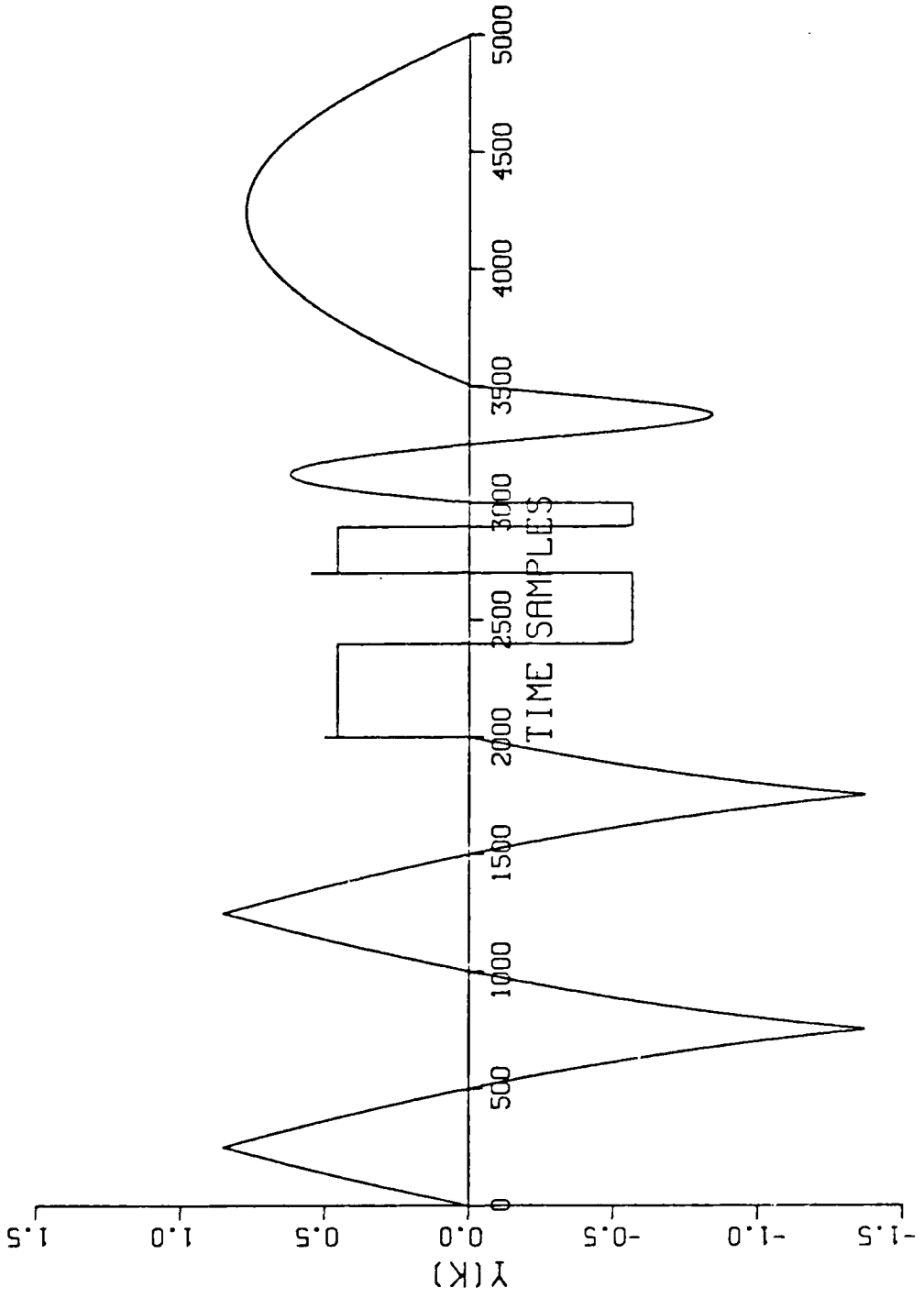


Figure 3.8 System II Output y(k)

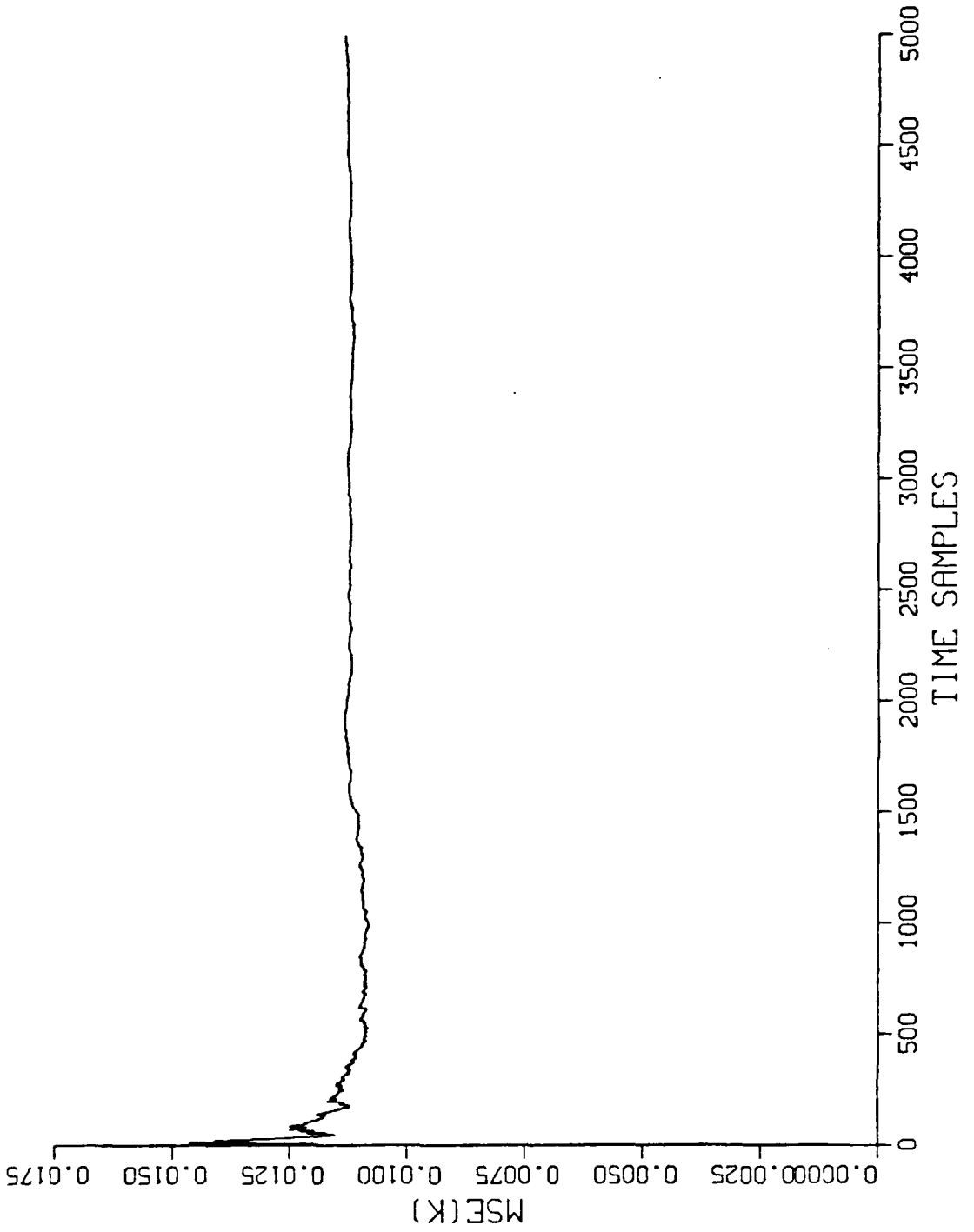


Figure 3.9 System II Mean-Square Error (Gaussian Noise Model)

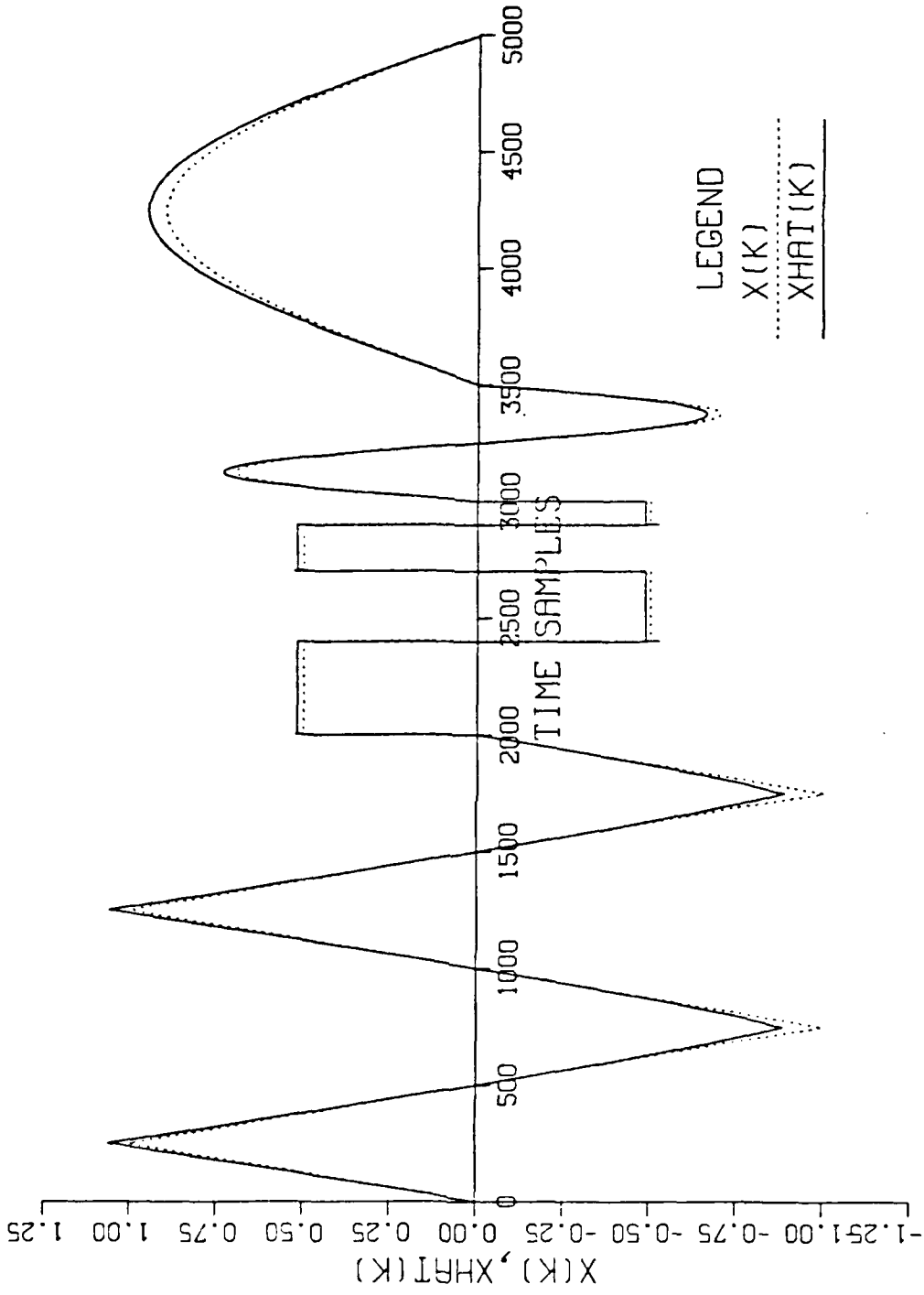


Figure 3.10 System II Comparison of x(k) and $\hat{x}(k)$ (Gaussian Noise Model)

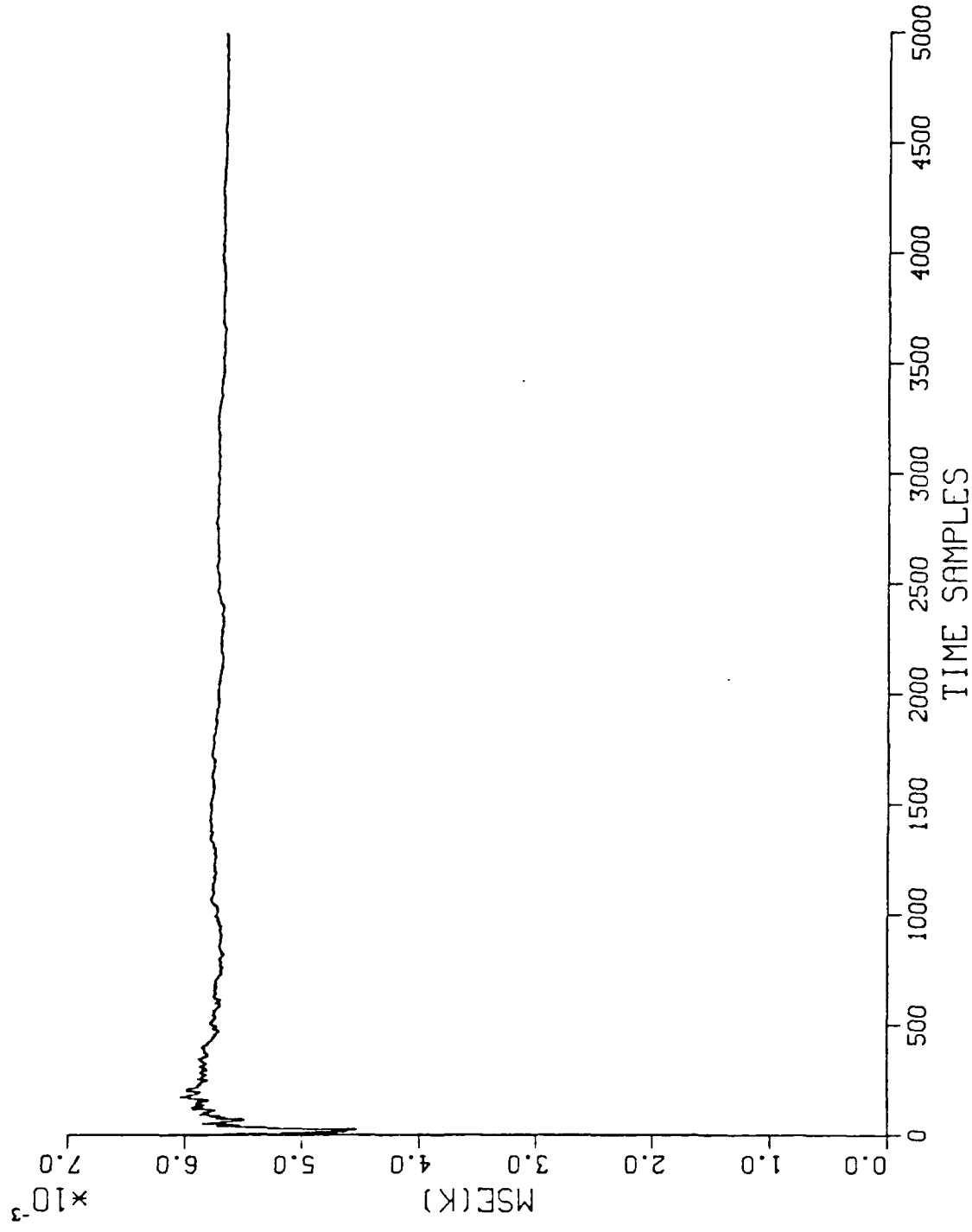


Figure 3.11 System II Mean-Square Error
(Uniform Noise Model)

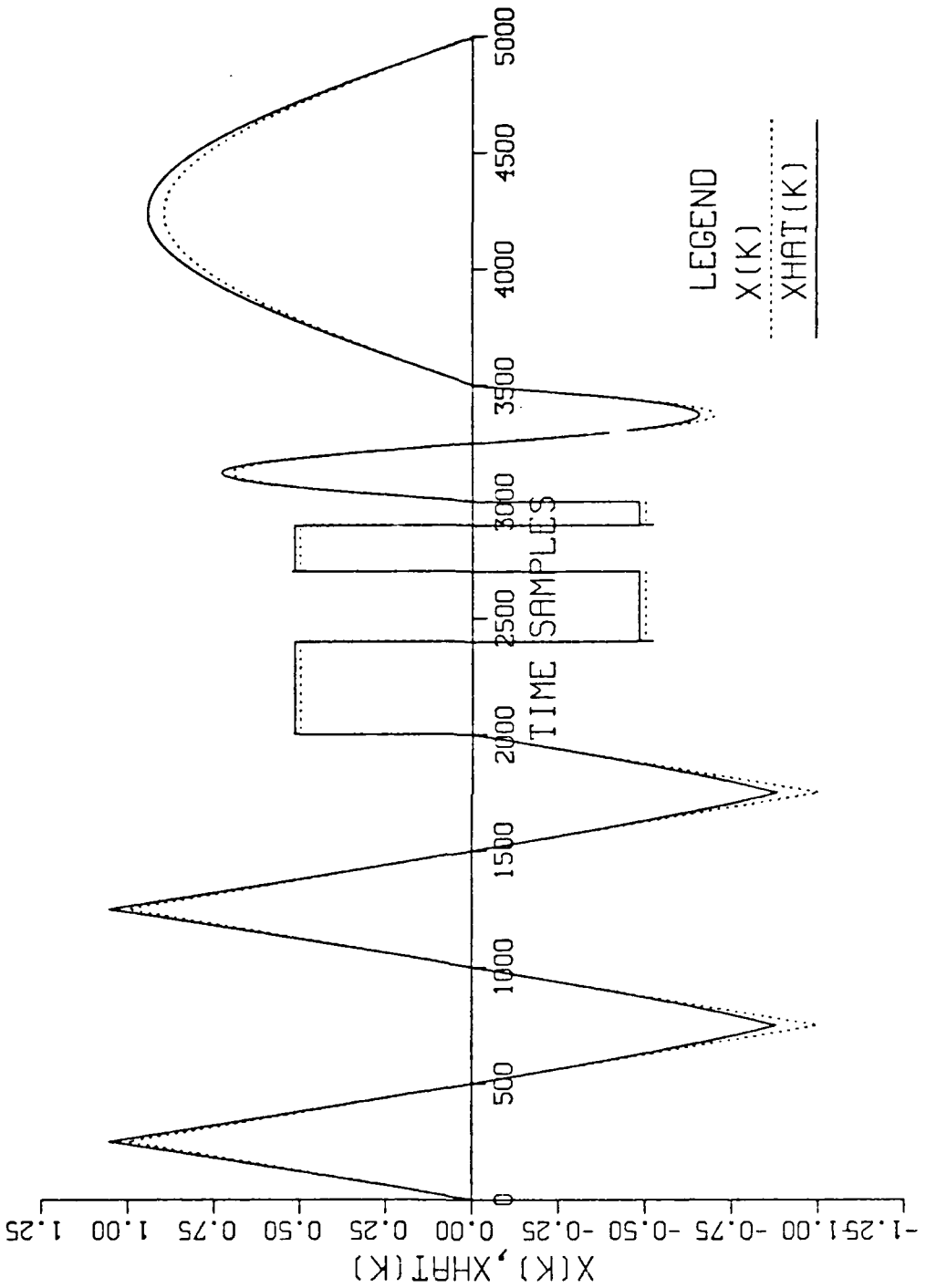


Figure 3.12 System II Comparison of x(k) and $\hat{x}(k)$ (Uniform Noise Model)

$$K = \begin{vmatrix} 0 & -.20 & .08 & 0 & -.04 & -.18 & 0 & 0 & 0 & 0 \\ 0 & 0 & -.20 & -.18 & 0 & .69 & .54 & .17 & .34 & -.42 \\ 0 & 0 & 0 & .12 & -.22 & -.39 & -.04 & .01 & .64 & .31 \\ 0 & 0 & 0 & 0 & -.18 & -.09 & -.37 & .60 & -.15 & .32 \\ 0 & 0 & 0 & 0 & 0 & .27 & .15 & -.67 & -.25 & -.13 \\ 0 & 0 & 0 & 0 & 0 & 0 & .57 & -.17 & -.37 & .49 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -.36 & -.37 & .50 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & .51 & -.38 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -.81 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}$$

Figure 3.13 provides the running average mean-square error plot. The output y(k) of the nonlinear system is shown in Figure 3.14. Figure 3.15 depicts the comparison between the system input x(k) and the lattice filter output $\hat{x}(k)$. The x(k) curve has the same shape as x(k), however, it is slightly offset. In an attempt to improve the approximation process, the number of realizations of the noise random process used to model the system was doubled from twenty-five to fifty and the simulation was repeated. Although several reflection coefficient values changed by .01, there was no perceptible improvement in the estimation of x(k).

d. System IV: y(k)=x(k) - (0.5+0.6y(k-1)+.08y(k-2))

This example consists of the linear System I modified by the addidtion of a constant bias term. Gaussian noise was used to model the system. The reflection coefficients determined for the first order model are:

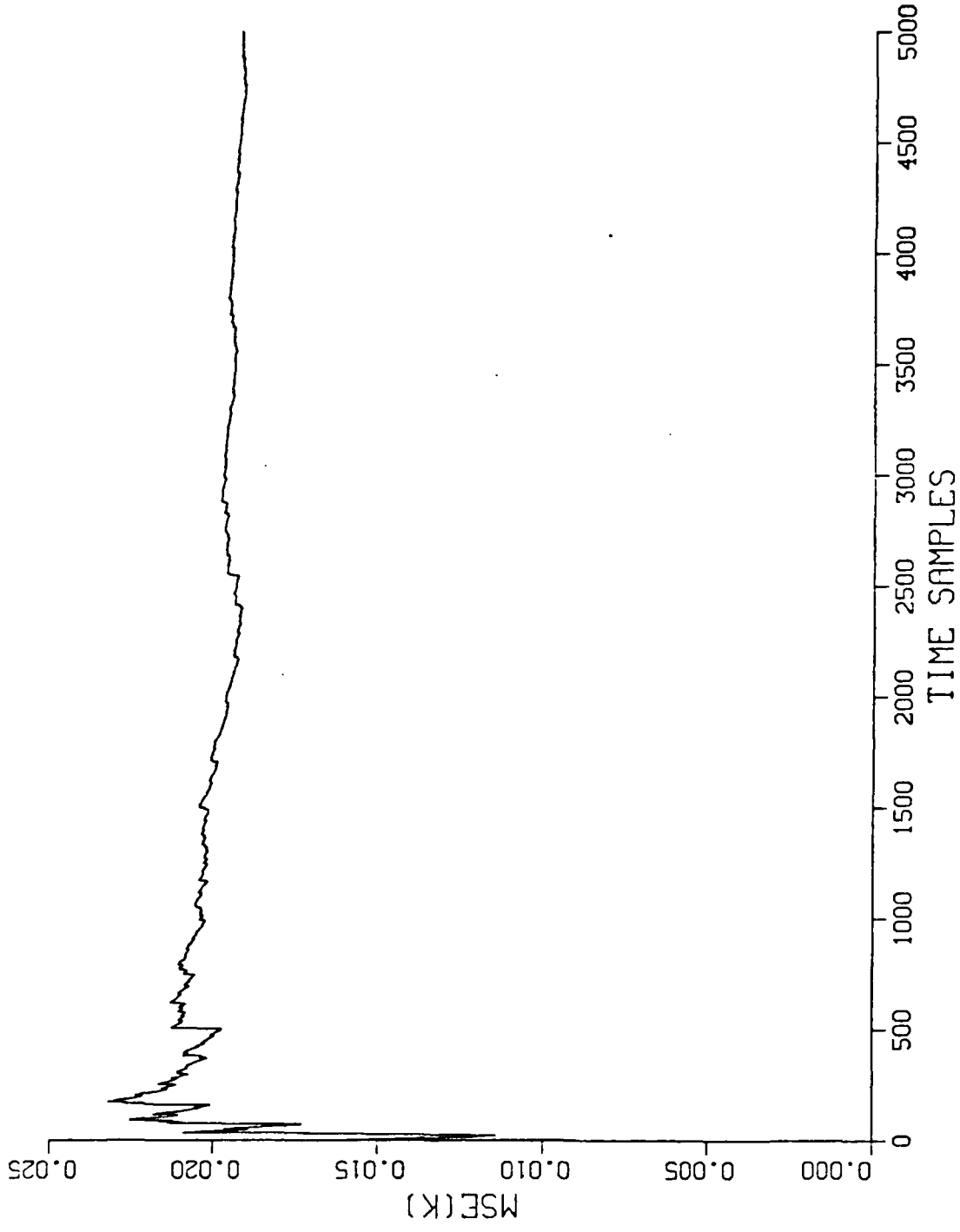


Figure 3.13 System III Mean-Square Error

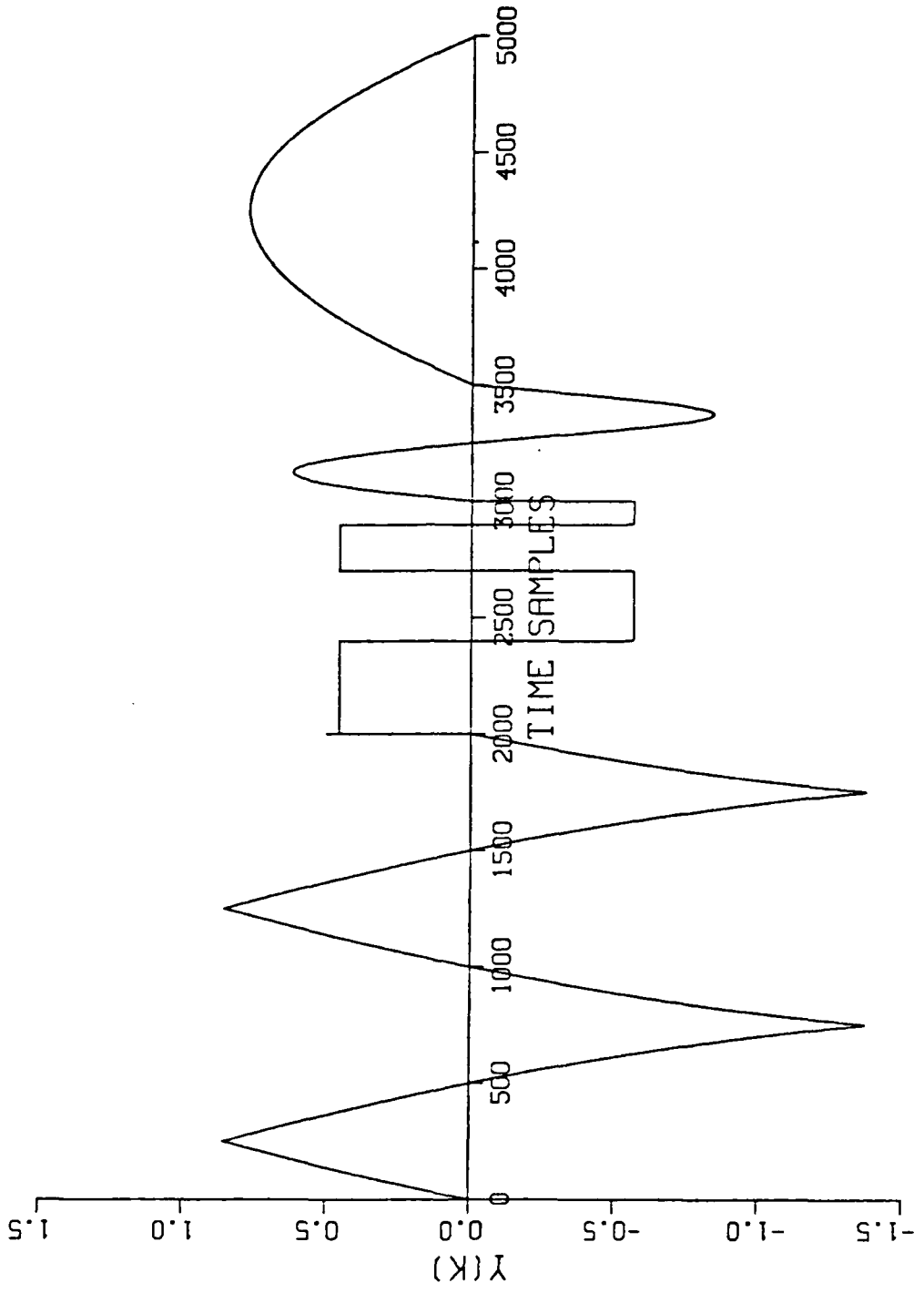


Figure 3.14 System III Output y(k)

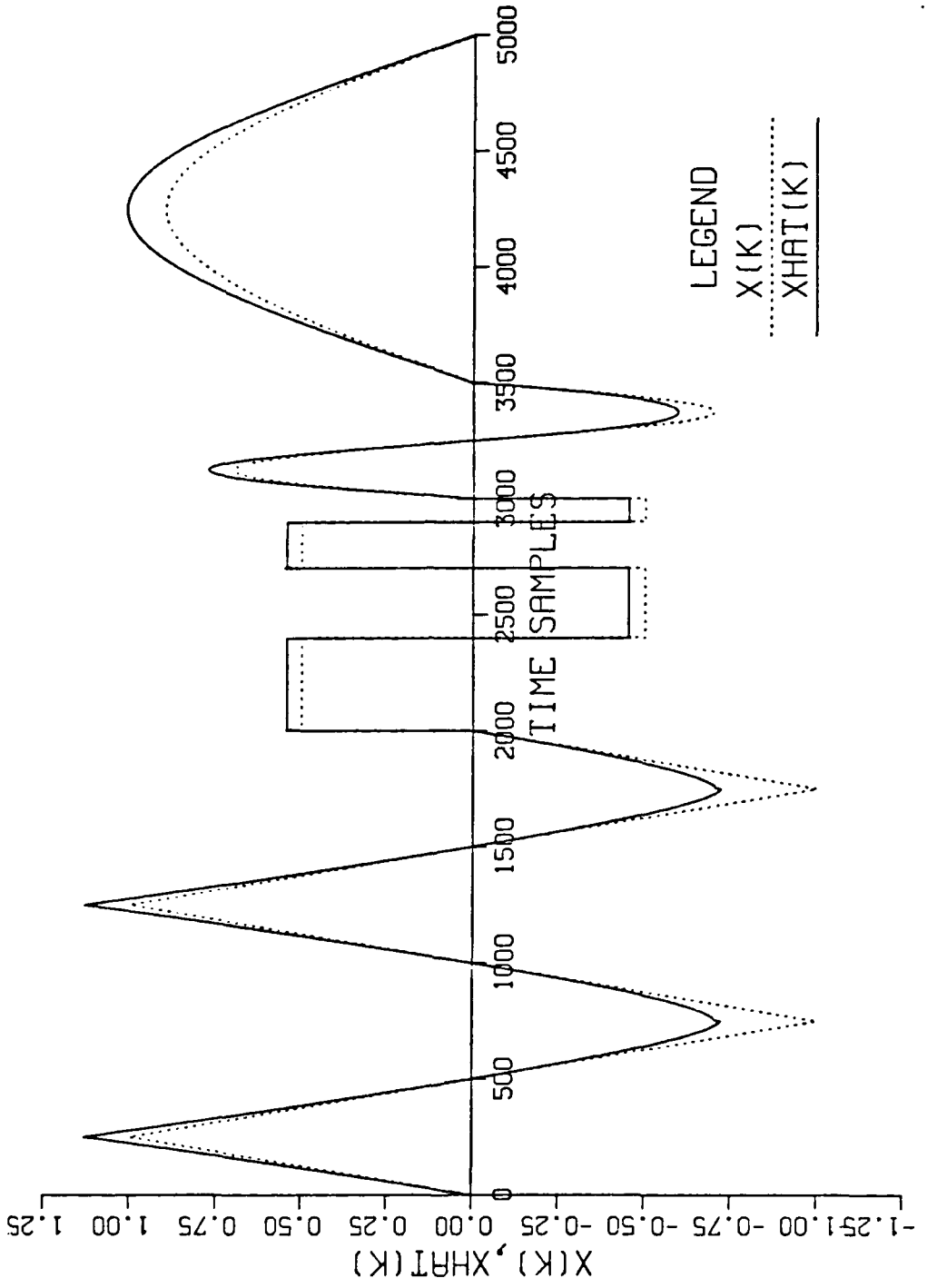


Figure 3.15 System III Comparison of x(k) and $\hat{x}(k)$

$$K = \begin{vmatrix} 0 & -.23 & -.55 & -.07 & 0 \\ 0 & 0 & -.23 & -.40 & -.43 \\ 0 & 0 & 0 & -.46 & .01 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix} .$$

This is basically the same matrix as obtained for System I, but modified by the addition of several terms which attempt to model the constant offset. (Note the reappearance of the values of -.55 and -.07 in the top row of the lattice matrix.) The running average mean-square error is shown in Figure 3.16. The system output y(k) is plotted in Figure 3.17. Figure 3.18 is a plot of the lattice output $\hat{x}(k)$ and the system input x(k). The $\hat{x}(k)$ curve is an excellent replica of the original input signal, but it is offset by a constant value. Although the small mean-square error evident in Figure 3.16 indicates that the lattice is a good inverse filter, the lattice was unable to completely remove the bias term. Increasing the order of the model (i.e., overmodeling the system) did not improve the results.

e. System V: $y(k)=x(k) - (5.0+0.6y(k-1)+.08y(k-2))$

In order to further investigate the constant offset nonlinearity, the example of System IV was repeated using a bias of 5.0 instead of 0.5. Again, Guassian white noise was used in determining the model's parameters. The reflection coefficients of the first order nonlinear lattice are given by:

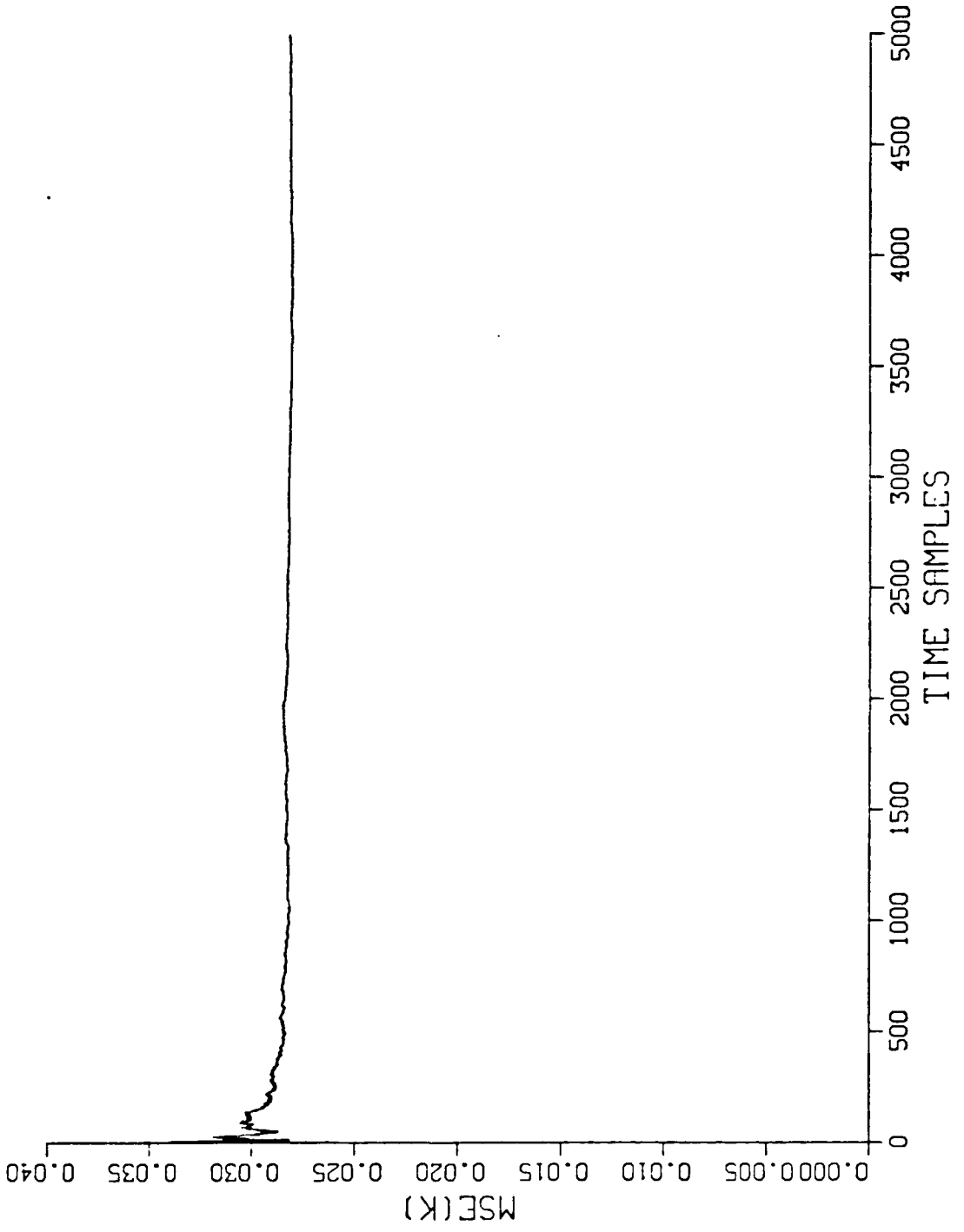


Figure 3.16 System IV Mean-Square Error

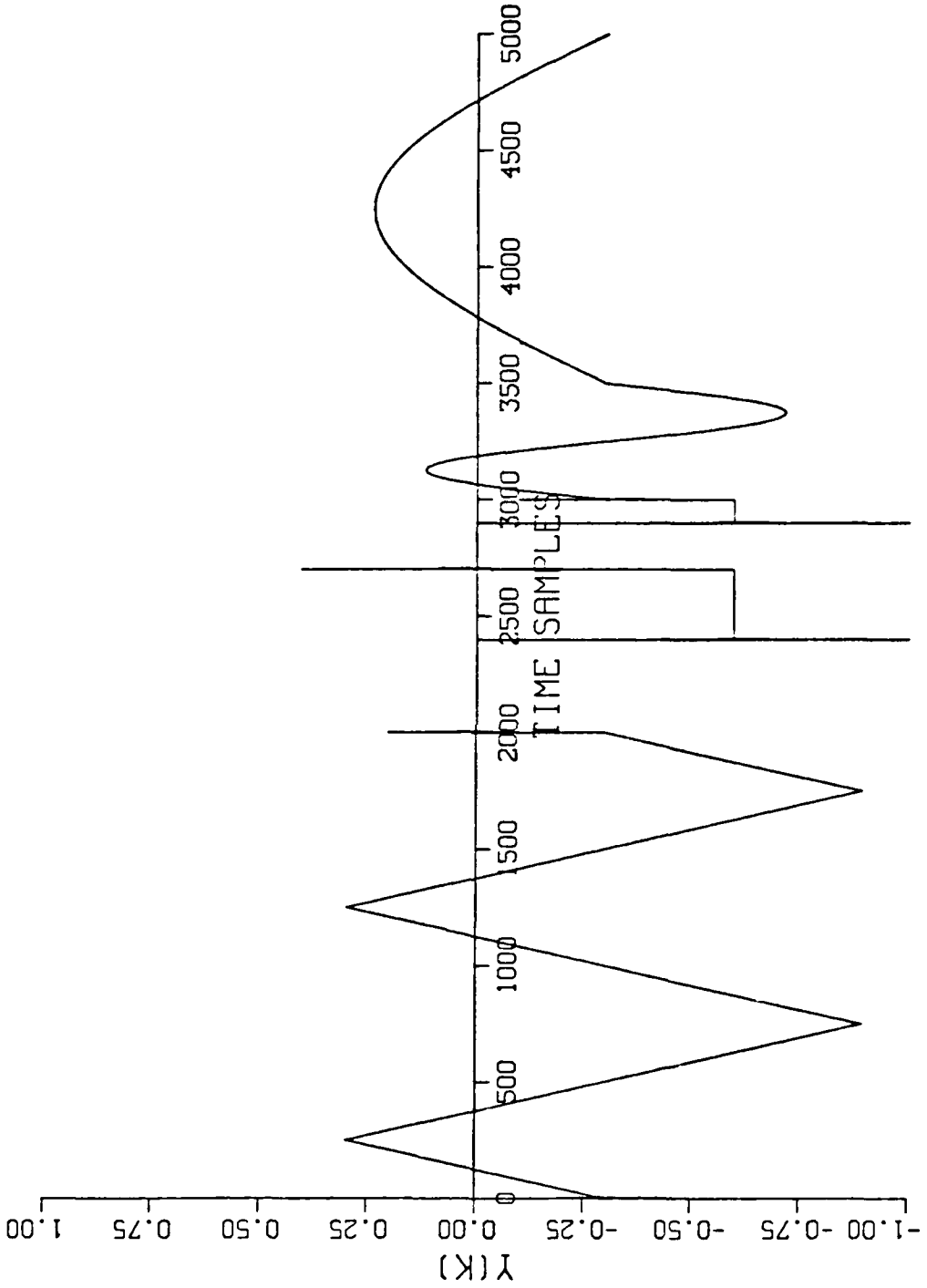


Figure 3.17 System IV Output y(k)

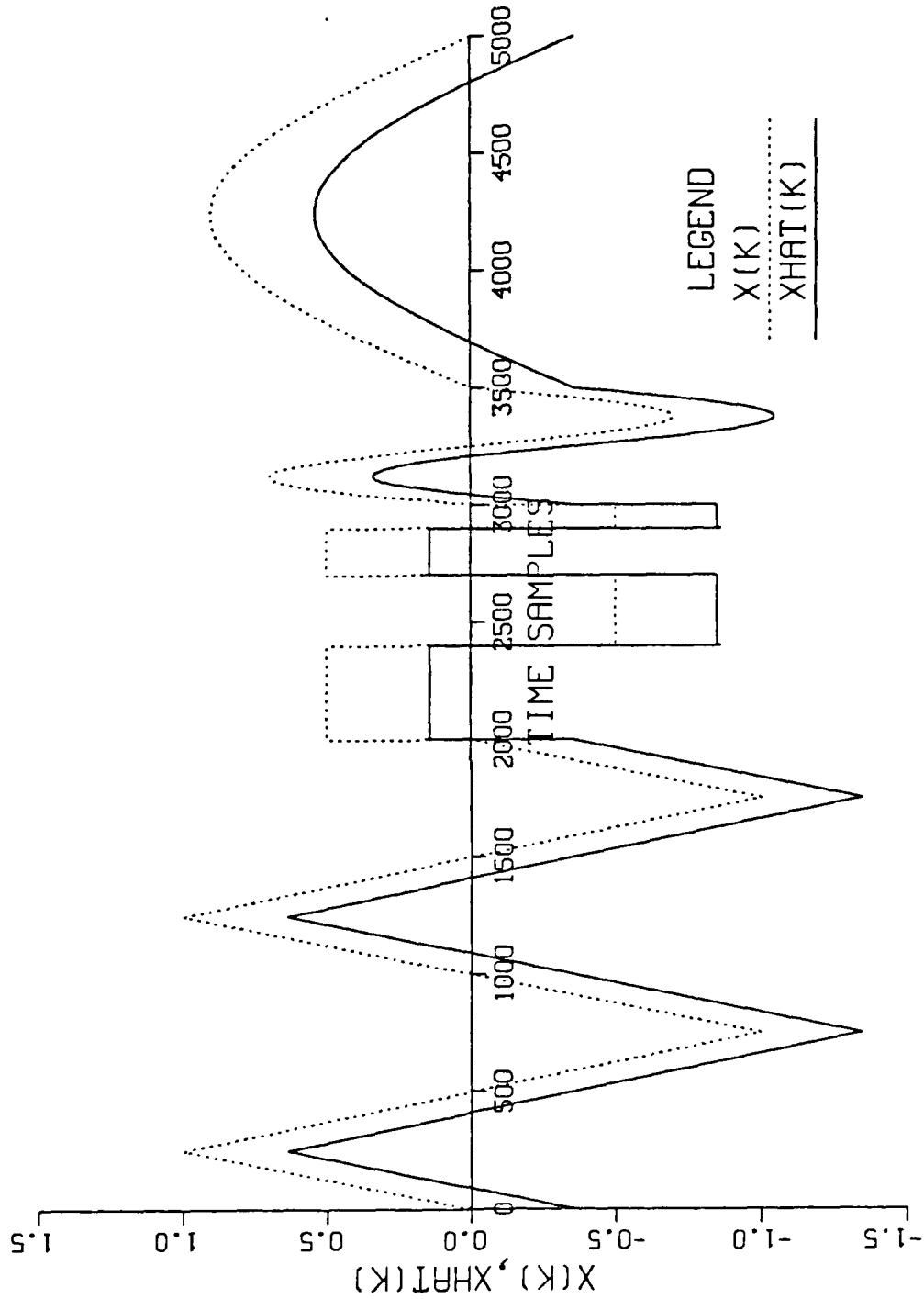


Figure 3.18 System IV Comparison of x(k) and $\hat{x}(k)$

$$K = \begin{vmatrix} 0 & -.92 & -.55 & -.07 & 0 \\ 0 & 0 & -.92 & -.86 & -.74 \\ 0 & 0 & 0 & .78 & -.75 \\ 0 & 0 & 0 & 0 & -.90 \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix} .$$

Again, the reflection coefficient values of -.55 and -.07 occur in the top row of the matrix as these elements apparently model the linear portion of the sytem. The running average mean-square error is displayed in Figure 3.19, and the output of the nonlinear system is shown in Figure 3.20. Figure 3.21 provides the comparison between the system input and the output of the inverse filter. As with System IV, the x(k) curve produced by the deconvolution process has the same shape as x(k), but is offset from the desired curve by a small constant value. Here, the system bias is 5.0, and the x(k) and $\hat{x}(k)$ curves differ by about 0.5. This is a relative improvement over the results obtained for System IV. System IV had a constant bias of only 0.5 but the $\hat{x}(k)$ curve was offset from the x(k) curve by approximately 0.4.

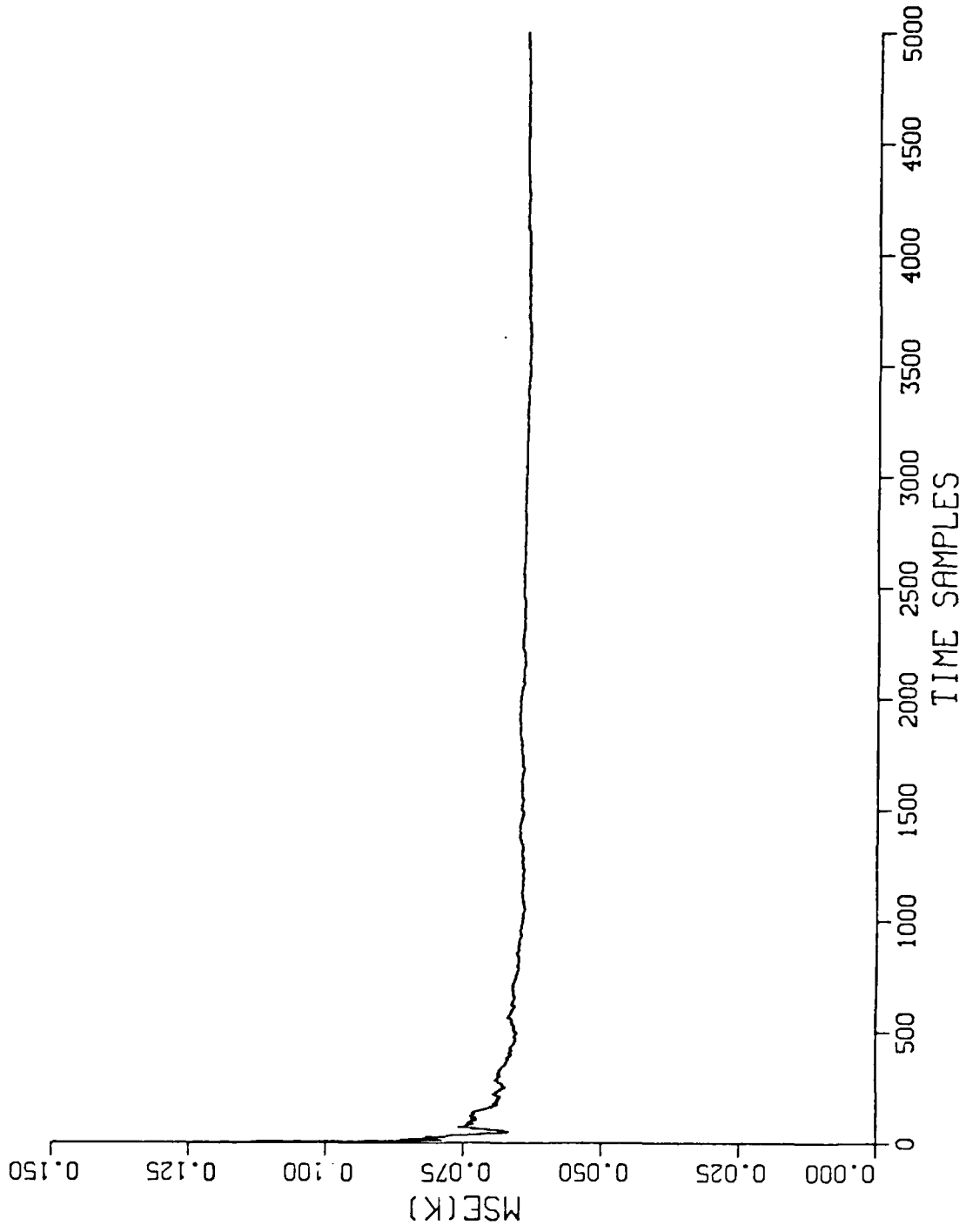


Figure 3.19 System V Mean-Square Error

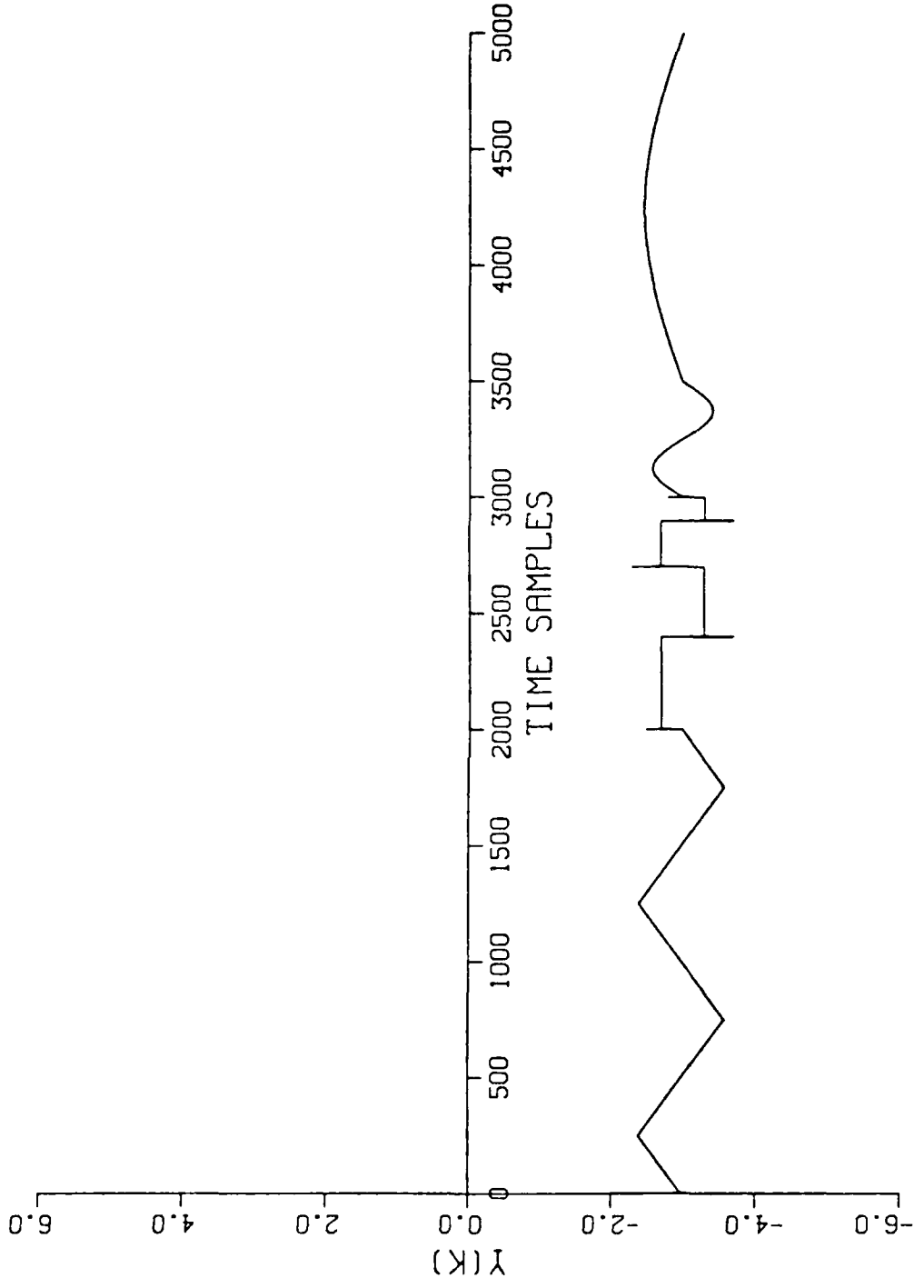


Figure 3.20 System V Output y(k)

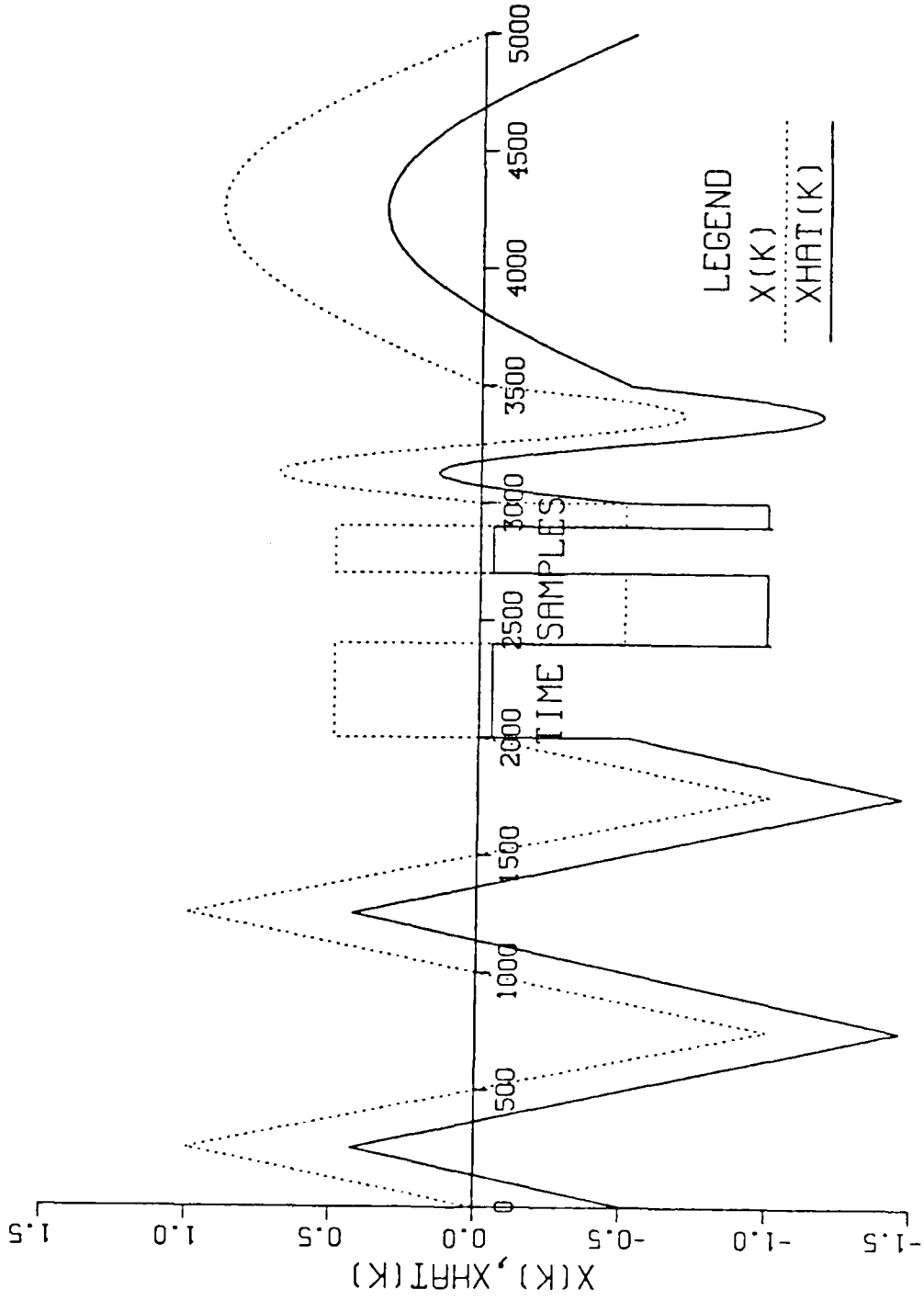


Figure 3.21 System V Comparison of x(k) and $\hat{x}(k)$

IV. CONCLUSION

In this thesis, several linear least-squares deconvolution, or inverse filtering, techniques were reviewed. Particular emphasis was placed on the lattice filter. It was shown that when a system is defined by an autoregressive, linear time-invariant model, this model can be transformed into an equivalent lattice filter representation. Furthermore, the resulting analysis lattice filter acts as a whitening filter, and is the inverse of the system's autoregressive transfer function. An example demonstrated the effectiveness of the lattice filter in a linear deconvolution application. Unfortunately, the linear lattice filter is unable to model nonlinear systems and, therefore, is not a viable nonlinear inverse filtering technique.

The discussion of the linear lattice filter led to the development of the generalized lattice filter, which in turn led to the derivation of the nonlinear lattice filter. The goal of this thesis was to implement the nonlinear lattice filter, and then apply it to the linear and nonlinear deconvolution problem. This was accomplished with generally very good results. It was shown that the nonlinear lattice filter was suitable for modeling

discrete autoregressive systems where the output y(n) consisted of a weighted sum of the cross-products of the terms $y^{p}(n-1)$ and $y^{q}(n-2)$, where the powers p and q can take on the values {0,1,2,3,4}. Examples were presented demonstrating the ability of the nonlinear lattice filter to effectively act as an inverse filter for both linear and nonlinear autoregressive systems.

APPENDIX A

LINEAR LATTICE FILTER FORTRAN PROGRAMS

```
C
C     ******************************************************************
C     *                                                                *
C     *   LT SCOT L. JOHNSON                                           *
C     *   LINEAR INVERSE                                               *
C     *   06 APRIL 1986                                                *
C     *                                                                *
C     ******************************************************************
C        THE PURPOSE OF THIS PROGRAM IS TO MODEL A LINEAR AUTOREGRESSIVE
C     SYSTEM BY THE EQUIVALENT ANALYSIS LATTICE FILTER.
C     THE SYSTEM'S TRANSFER FUNCTION IS GIVEN BY H(Z) = 1/A(Z)
C     WHERE A(Z) = (Z**2 + A1*Z + A2)/Z**2.
C          THE LATTICE FILTER REFLECTION COEFFICIENTS ARE DETERMINED
C     BY DRIVING H(Z) WITH ZERO MEAN, UNITY VARIANCE GAUSSIAN WHITE
C     NOISE. THE OUTPUT IS PROCESSED BY SUBROUTINE ATOCOR WHICH COMPUTES
C     THE COMPONENTS OF THE SAMPLED AUTOCORRELATION MATRIX R. SUBROUTINE
C     LEVIN IMPLEMENTS THE LEVINSON ALGORITHM AND EVALUATES THE LATTICE
C     FILTER COEFFICIENTS FROM THE AUTOCORRELATION MATRIX. FINALLY,
C     SUBROUTINE LATICE IMPLEMENTS THE ANALYSIS STRUCTURE OF THE
C     LATTICE FILTER WHICH IS EQUIVALENT TO THE INVERSE FILTER H(Z).
C          NOW WHEN H(Z) IS DRIVEN BY AN UNKNOWN  SIGNAL X(N), THE
C     SYSTEM OUTPUT Y(N) CAN BE FED INTO THE PREVIOUSLY DETERMINED
C     LATTICE FILTER; SINCE THE LATTICE FILTER IS THE INVERSE OF H(Z)
C     THE LATTICE FILTER OUTPUT SHOULD BE A GOOD ESTIMATE OF X(N).
C
C     ***VARIABLE DEFINITIONS****
C     A1, A2 = COEFFICIENTS OF THE PREDICTION ERROR POLYNOMIAL A(Z)
C     DSEED,DSEED1 = SEED VALUES USED BY THE IMSL WHITE GAUSSIAN NOISE
C                    GENERATOR FUNCTION GGNQG
C     E = VECTOR OF MEAN-SQUARED PREDICTION ERRORS E(0),E(1),...E(ORDER)
C     ESUM = VECTOR USED IN DETERMINING AVERAGE E
C     GAMMA = VECTOR OF LATTICE REFLECTION COEFFICIENTS. GAMMA(I) IS
C             THE REFLECTION COEFFICIENT OF THE I'TH LATTICE STAGE.
C     GAMSUM = VECTOR USED IN DETERMING AVERAGE GAMMA
C     GRAFP = REAL VALUE WHICH DEFINES LENGTH OF X-AXIS FOR PLOTTING
C     L = LOWER-TRIANGULAR MATRIX WHOSE ROWS ARE THE REVERSE OF ALL THE
C       PREDICTION ERROR FILTERS FROM ORDER ZERO TO THE HIGHEST ORDER
C     LSUM = MATRIX USED IN DETERMINING THE AVERAGE L MATRIX
C     MSE = MEAN-SQUARE ERROR BETWEEN X(N) AND XHAT(N)
C     MSEMAX = MAXIMUM VALUE OF MSE
C     MSESUM = RUNNING SUM USED IN CALCULATING MSE
C     NINDEX = ARRAY OF REAL NUMBERS USED IN DISSPLA PLOTTING ROUTINES
C     NOISE = ARRAY OF MEASUREMENT NOISE ADDED TO OUTPUT OF H(Z)
C     NUMPTS = NUMBER OF POINTS IN INPUT NOISE SEQUENCE
C     ORDER = ORDER OF THE LATTICE FILTER
C     ORDERP = ORDER + 1
C     PLTPTS = NUMBER OF POINTS USED IN PLOTTING ROUTINES
C     R = VECTOR OF AUTOCORRELATION LAGS R(0),R(1),...R(ORDER)
C     RMAX = MAXIMUM MAGNITUDE OF ELEMENTS OF R; TO BE USED IN PLOTTING
C     STDDEV = STANDARD DEVIATION OF THE MEASUREMENT NOISE
C     TRIAL = NUMBER OF REALIZATIONS OF THE WHITE GAUSSIAN NOISE RANDOM
C             PROCESS USED IN MODELING THE SYSTEM H(Z)
C     X = INPUT SEQUENCE INTO THE SYSTEM H(Z)
C     XMAX, XMIN = RANGE OF X VALUES USED IN DISSPLA PLOTTING ROUTINES
C     XHAT = ESTIMATE OF X; OUTPUT OF THE ANALYSIS LATTICE FILTER A(Z)
C     Y = OUTPUT SEQUENCE OF H(Z)
C     YMAX, YMIN = RANGE OF Y VALUES USED IN DISSPLA PLOTTING ROUTINES
C
C     ***VARIABLE DECLARATIONS****
      INTEGER I,N,K,NUMPTS,ORDER,ORDERP,PLTPTS,TRIAL
```

```
      REAL*4 X(5000),Y(5000),A1,A2,XMIN,XMAX,YMAX,YMIN,NINDEX(5000)
      REAL*4 GAMMA(10),GAMSUM(10),R(10),L(10,10),E(10),RMAX,GRAFP
      REAL*4 XHAT(5000),LSUM(10,10),ESUM(10),NOISE(5000),STDDEV
      REAL*4 MSE(5000),MSESUM,MSEPLT
      REAL*8 DSEED,DSEED1
C
C
C     DEFINE THE LENGTH OF THE TIME SEQUENCES
      NUMPTS = 5000
      PLTPTS = 2000
      GRAFP = FLOAT(PLTPTS+1)
C
C     DEFINE THE ORDER OF THE LATTICE FILTER AND THE A(Z) COEFFICIENTS
      ORDER=2
      ORDERP=ORDER+1
      A1 = 0.6
      A2 = 0.08
C     INITIALIZE VARIABLES
      DO 6 I=1,ORDERP
            GAMMA(I) = 0.
            GAMSUM(I) = 0.
            ESUM(I) = 0.
            R(I) = 0.
            DO 4 K=1,ORDERP
               LSUM(I,K) = 0.
4           CONTINUE
6     CONTINUE
C
      XMAX = 0.
      YMAX = 0.
      MSESUM = 0.
      MSEMAX = 0.
C     DEFINE THE SEEDS FOR THE NOISE SIGNALS, THE NUMBER OF NOISE
C     REALIZATIONS USED IN MODELING H(Z), AND THE STANDARD DEVIATION OF
C     THE MEASUREMENT NOISE
      DSEED = 1243073.5D0
      DSEED1 = 724389.4D0
      TRIAL = 25
      STDDEV = 0.05
C
      DO 25 I=1,TRIAL
           DSEED = DSEED/DFLOAT(I)
           DSEED1 = DSEED1/DFLOAT(I)
           DO 10 K=1,NUMPTS
              X(K) = GGNQF(DSEED)
10         CONTINUE
           DO 11 K=1,NUMPTS
              NOISE(K) = STDDEV * GGNQF(DSEED1)
11         CONTINUE
C
           Y(1) = X(1) + NOISE(1)
           Y(2) = X(2) - A1*Y(1) + NOISE(2)
           DO 12 K = 3,NUMPTS
              Y(K) = X(K) - (A1*Y(K-1) + A2*Y(K-2)) + NOISE(K)
12         CONTINUE
C
C          DETERMINE THE AUTOCORRELATION VECTOR OF THE SEQUENCE Y(N)
           CALL ATOCOR(NUMPTS,Y,ORDER,R,RMAX)
C          DETERMINE THE L MATRIX AND E VECTOR
           CALL LEVIN(ORDER,R,L,E)
           DO 18 K=1,ORDER
              GAMMA(K)=-1*L(K+1,1)
              GAMSUM(K) = GAMSUM(K) + GAMMA(K)
18         CONTINUE
C
           DO 22 N=1,ORDERP
              ESUM(N) = ESUM(N) + E(N)
              DO 20 K=1,ORDERP
                 LSUM(N,K) = LSUM(N,K) + L(N,K)
```

```
20            CONTINUE
22         CONTINUE
25      CONTINUE
C
C       CALCULATE THE AVERAGE VALUES OF E, L, AND GAMMA
         DO 28 N=1,ORDERP
            E(N) = ESUM(N)/FLOAT(TRIAL)
            DO 26 K=1,ORDERP
               L(N,K) = LSUM(N,K)/FLOAT(TRIAL)
26         CONTINUE
28      CONTINUE
         DO 30 K=1,ORDER
            GAMMA(K) = GAMSUM(K)/FLOAT(TRIAL)
30      CONTINUE
C
C       WITH INPUT Y AND LATTICE PARAMETERS GAMMA, DETERMINE THE
C          OUTPUT OF THE ANALYSIS MODEL LATTICE FILTER
C       CALL LATICE(NUMPTS,ORDER,GAMMA,Y,XHAT)
C
C            PRINT RESULTS
C
C       PRINT THE R,E, AND GAMMA VECTORS
         WRITE(8,35)
35      FORMAT(T1,'1',T4,'N',T10,'R(N)',T20,'E(N)',T30,'GAMMA(N)')
         NULL=0
         WRITE(8,40) NULL,R(1),E(1)
40      FORMAT(T1,'0',T4,I2,T10,3F10.4)
         DO 45 K=2,ORDERP
            N=K-1
            WRITE(8,40) N,R(K),E(K),GAMMA(N)
45      CONTINUE
C
C       PRINT THE L(I,J) MATRIX
         WRITE(8,48)
48      FORMAT(T1,'1','L(I,J)=')
         DO 55 I=1,ORDERP
            WRITE(8,50) (L(I,J),J=1,ORDERP)
50          FORMAT(T1,'0',10(F10.4,4X))
55      CONTINUE
C
C
C       DO THE DECONVOLUTION PROBLEM. FIRST DEFINE THE DETERMINISTIC
C       INPUT X(N), THEN DETERMINE THE SYSTEM OUTPUT.
         DO 85 K=1,PLTPTS
            X(K) = 1.0 * SIN(0.0126*FLOAT(K))
85      CONTINUE
C
         Y(1) = X(1) + NOISE(1)
         Y(2) = X(2) - A1*Y(1) + NOISE(2)
         DO 90 K=3,PLTPTS
            Y(K) = X(K) - (A1*Y(K-1) + A2*Y(K-2)) + NOISE(K)
90      CONTINUE
C
C       INPUT Y(N) INTO THE LATTICE TO RECOVER THE ESTIMATE OF X(N)
         CALL LATICE(PLTPTS,ORDER,GAMMA,Y,XHAT)
C
C       CALCULATE THE MEAN-SQUARE ERROR (MSE) BETWEEN X(N) AND XHAT(N)
         DO 93 K=1,PLTPTS
            MSESUM = (X(K)-XHAT(K))*(X(K)-XHAT(K)) + MSESUM
            MSE(K) = SQRT(MSESUM/FLOAT(K))
            IF(MSE(K).GT.MSEMAX) MSEMAX = MSE(K)
93      CONTINUE
C
C       DEFINE PLOTTING PARAMETERS
         DO 95 K= 1,PLTPTS
            NINDEX(K) = FLOAT(K-1)
            IF(ABS(X(K)).GT.XMAX) XMAX = ABS(X(K))
            IF(ABS(XHAT(K)).GT.XMAX) XMAX = ABS(XHAT(K))
            IF(ABS(Y(K)).GT.YMAX) YMAX = ABS(Y(K))
```

```
95    CONTINUE
        XMIN = -1.0 * XMAX
        YMIN = -1.0 * YMAX
C
C     **** DISSPLA PLOTTING ROUTINES ****
C
C     PLOT THE SYSTEM INPUT AND THE LATTICE OUTPUT
C     CALL TEK618
        CALL COMPRS
        CALL RESET('ALL')
        CALL PAGE(8.0,6.5)
        CALL XINTAX
        CALL AREA2D(6.75,5.0)
        CALL XNAME('TIME SAMPLES$',100)
        CALL YNAME('X(N)$',100)
        CALL CROSS
        CALL GRAF(0.,500.,GRAFP,XMIN,'SCALE',XMAX)
        CALL CURVE(NINDEX,X,PLTPTS,0)
        CALL ENDPL(0)
C
C     PLOT THE SYSTEM OUTPUT Y(N)
        CALL NOBRDR
        CALL AREA2D(6.75,5.0)
        CALL XNAME('TIME SAMPLES$',100)
        CALL YNAME('Y(N)$',100)
C     CALL HEADIN('Y(N) = X(N) - (0.6*Y(N-1)+0.08*Y(N-2))$',100,1.5,1)
        CALL HEADIN('Y(N)=X(N)-(0.6*Y(N-1)+0.08*Y(N-2))+NOISE$',100,1.5,2)
        CALL HEADIN('NOISE = N(0,0.0025)$',100,1.5,2)
        CALL CROSS
        CALL GRAF(0.,500.,GRAFP,YMIN,'SCALE',YMAX)
        CALL CURVE(NINDEX,Y,PLTPTS,0)
        CALL ENDPL(0)
C
C     PLOT LATTICE FILTER OUTPUT, XHAT(N)
        CALL AREA2D(6.75,5.0)
        CALL XNAME('TIME SAMPLES$',100)
        CALL YNAME('XHAT(N)$',100)
        CALL CROSS
        CALL GRAF(0.,500.,GRAFP,XMIN,'SCALE',XMAX)
        CALL CURVE(NINDEX,XHAT,PLTPTS,0)
        CALL ENDPL(0)
C
C     PLOT THE MEAN-SQUARE ERROR BETWEEN X(N) AND XHAT(N)
        CALL AREA2D(6.75,5.0)
        CALL XNAME('TIME SAMPLES$',100)
        CALL YNAME('MSE(N)$',100)
        CALL CROSS
C     CALL GRAF(0.,500.,GRAFP,0.,'SCALE',MSEMAX)
        CALL GRAF(0.,500.,GRAFP,0.,'SCALE',0.08)
        CALL CURVE(NINDEX,MSE,PLTPTS,0)
        CALL ENDPL(0)
        CALL DONEPL
C
        STOP
        END
C
C*************************************************************************
C
C     SUBROUTINE ATOCOR
C
C     GIVEN A TIME SEQUENCE Y(N), THIS PROGRAM CALCULATES THE SAMPLED
C     AUTOCORRELATION MATRIX TERMS R(0),R(1),...,R(ORDER).
C
C     WRITTEN 06 APRIL 1986
C
C*************************************************************************
      SUBROUTINE ATOCOR(NUMPTS,Y,ORDER,R,RMAX)
C     VARIABLE DEFINITIONS:
```

```
C           Y(5000)= INPUT SEQUENCE
C           R(10) = AUTOCORRELATION MATRIX
C           NUMPTS = NUMBER OF POINTS IN THE SEQUENCE Y(N)
C           ORDER = MAXIMUM LAG FOR WHICH R IS EVALUATED
C           ORDERP = ORDER+1;THE LENGTH OF THE R VECTOR
C           SUM= RUNNING TOTAL OF PRODUCTS FOR A GIVEN LAG
C           RMAX= MAXIMUM VALUE OF R; USED FOR DISSPLA PLOTING
C
        INTEGER NUMPTS,I,J,K,N,ORDER,ORDERP
        REAL Y(5000),R(10),SUM,RMAX
        RMAX=0.
        ORDERP=ORDER+1
        DO 75 K=1,ORDERP
              SUM=0.
              N=NUMPTS-K+1
              DO 70 J=1,N
                    SUM=SUM+(Y(K+J-1)*Y(J))
70            CONTINUE
              R(K)=SUM/NUMPTS
              IF(ABS(R(K)).GT.RMAX) RMAX=ABS(R(K))
75      CONTINUE
C
        RETURN
        END
C
C
C*************************************************************************
C
C       SUBROUTINE LEVIN
C
C       THIS SUBROUTINE IMPLEMENTS LEVINSON'S ALGORITHM. IT GENERATES ALL
C       THE PREDICTION ERROR FILTERS UP TO A GIVEN ORDER, FROM THE
C       AUTOCORRELATION LAGS.
C
C       BASED ON A PROGRAM WRITTEN BY S.J. ORFANIDIS (REF. 1:P. 333)
C
C       MODIFIED 06 APRIL 1986
C
C*************************************************************************
        SUBROUTINE LEVIN(ORDER,R,L,E)
C       ***VARIABLE DEFINITIONS****
C
C       ORDER = ORDER OF LATTICE FILTER
C       R = VECTOR OF AUTOCORRELATION LAGS R(0),R(1),...,R(ORDER)
C       L = UNIT LOWER-TRIANGULAR MATRIX.  ITS I-TH ROW HOLDS THE I-TH
C            PREDICTION-ERROR FILTER IN REVERSE ORDER. ITS FIRST COLUMN
C            HOLDS THE NEGATIVES OF THE REFLECTION COEFFICIENTS,
C            GAMMA(I) = -L(I+1,1) FOR I=1,2,...,ORDER
C       E = VECTOR OF PREDICTION ERRORS E(0),E(1),...,E(ORDER)
C       THE MATRIX L AND THE DIAGONAL MATRIX D FORMED BY THE E'S DEFINE
C       A UL CHOLESKY FACTORIZATION OF THE INVERSE OF THE AUTOCORRELATION
C       MATRIX:  R INVERSE = L TRANSPOSE * D INVERSE * L
C
C       ***VARIABLE DECLARATIONS****
        REAL R(10),E(10),L(10,10),GAP,GAMMA
        INTEGER I,IPLUS,IMINUS,J,ORDER,ORDERP
        ORDERP=ORDER+1
C       SET THE UPPER TRIANGLE OF THE L MATRIX TO ZERO
        DO 60 I=1,ORDER
              IPLUS=I+1
              IMINUS=I-1
              DO 60 J=IPLUS,ORDERP
                    L(I,J)=0.
60      CONTINUE
C
        L(1,1)=1.
        L(2,2)=1.0
        L(2,1)=-R(2)/R(1)
        E(1)=R(1)
```

```
        E(2)=E(1)*(1.-L(2,1)**2)
        DO 68 I=3,ORDERP
              GAP=0.
              IMINUS=I-1
              DO 63 K=1,IMINUS
                    GAP=GAP+R(K+1)*L(I-1,K)
63            CONTINUE
              GAMMA=GAP/E(I-1)
              L(I,1)=-1*GAMMA
              DO 64 K=2,IMINUS
                    L(I,K)=L(I-1,K-1)-GAMMA*L(I-1,I-K)
64            CONTINUE
              L(I,I)=1.0
              E(I)=E(I-1)*(1-GAMMA**2)
68    CONTINUE
C
        RETURN
        END
C
C*************************************************************************
C
C     SUBROUTINE LATICE
C
C     THIS PROGRAM IMPLEMENTS A SINGLE CHANNEL LATTICE STRUCTURE.
C     WHEN GIVEN THE LATTICE COEFFICIENTS AND THE INPUT SEQUENCE,
C     THE PROGRAM DETERMINES THE OUTPUT SEQUENCE.
C
C     WRITTEN 06 APRIL 1986
C
C*************************************************************************
C
        SUBROUTINE LATICE(NUMPTS,ORDER,GAMMA,Y,OUTPUT)
C     ***VARIABLE DEFINITIONS****
C     NUMPTS= NUMBER OF POINTS IN THE SEQUENCES; MAX IS 5000
C     ORDER= ORDER OF THE LATTICE; MAX IS 9
C     GAMMA(ORDER)= LATTICE COEFFICENT ARRAY
C     F = FORWARD ERROR
C     B = BACKWARD ERROR
C     DELAY(ORDER)= ARRAY OF DELAYED BACKWARD ERROR SIGNALS
C     TEMP(ORDER) = ARRAY WHICH TEMPORARILY HOLDS THE BACKWARD ERROR
C     Y(NUMPTS)=INPUT DATA ARRAY
C     OUTPUT(NUMPTS)=ARRAY OF LATTICE OUTPUT DATA
C
C     ***VARIABLE DECLARATIONS****
        INTEGER NUMPTS,ORDER,I,K,M
        REAL GAMMA(10),F,B,DELAY(10),TEMP(10),Y(5000),OUTPUT(5000)
C     INITIALIZE ARRAYS
        DO 80 I=1,ORDER
              DELAY(I)=0.
              TEMP(I)=0.
80    CONTINUE
C
C     DO TIME ITERATION
        DO 88 K=1,NUMPTS
              F=Y(K)
              B=Y(K)
C           FOR EACH TIME INSTANT,RECURSIVELY INCREASE THE LATTICE ORDER
              DO 85 M=1,ORDER
                    TEMP(M)=B
                    B=DELAY(M)-(GAMMA(M)*F)
                    F=F-(GAMMA(M)*DELAY(M))
                    DELAY(M)=TEMP(M)
85            CONTINUE
              OUTPUT(K)=F
88    CONTINUE
        RETURN
        END
```

APPENDIX B

NONLINEAR LATTICE FILTER FORTRAN PROGRAMS

```
C*************************************************************************
C                                                                        *
C     PROGRAM NLMAIN                                                     *
C                                                                        *
C     THIS PROGRAM DEFINES THE SYSTEM PARAMETERS AND THE SYSTEM'S        *
C     INPUTS AND OUTPUTS.  IT CALLS SUBROUTINES TO DETERMINE THE         *
C     THE CORRESPONDING NONLINEAR ANALYSIS LATTICE MODEL.                *
C                                                                        *
C     WRITTEN 30 APRIL 1986                                              *
C                                                                        *
C*************************************************************************
C
C     THIS PROGRAM INPUTS A WHITE NOISE SEQUENCE X(K) INTO A AUTOREGRES-
C     SIVE, NONLINEAR SYSTEM H(Z).  THE SYSTEM'S OUTPUT, Y(K), IS
C     PROCESSED TO DETERMINE THE AUTOCORRELATION MATRIX AND THE NONLINEA
C     LATTICE'S REFLECTION COEFFICIENT MATRIX.  SINCE THE OUTPUT OF THE
C     LATTICE IS WHITE NOISE, THE LATTICE IS EQUIVALENT TO THE INVERSE
C     OF THE SYSTEM.
C
C     *** VARIABLE DECLARATIONS****
C     DSEED = SEED USED BY THE IMSL WHITE GAUSSIAN NOISE FUNCTION GGNQF
C     GRAFP = MAX X-AXIS VALUE FOR X AND Y GRAPHS
C     GRAFN = MAX X-AXIS VALUE FOR MSE GRAPH
C     H = ARRAY OF AUTOREGRESSIVE PARAMETERS THAT DEFINE THE SYSTEM
C     IY = SEED USED BY THE UNIFORM WHITE NOISE FUNCTION URAND
C     MN = N * N
C     MNP1 = MN + 1; NUMBER OF ROWS IN THE NONLINEAR LATTICE
C     MSE = MEAN SQUARE ERROR BETWEEN THE INPUT NOISE SIGNAL AND THE
C           FORWARD ERROR SIGNAL FROM THE TOP ROW OF THE LATTICE
C     MSESUM = RUNNING TOTAL USED IN CALCULATION MSE
C     MSEMAX = MAXIMAUM VALUE OF MSE FOR USE IN PLOTTING
C     MSEPLT = REAL*4 VALUES OF MSE USED IN DISSPLA PLOTTING
C     NORM = ARRAY OF FACTORS THAT NORMALIZE THE LATTICE INPUTS
C     N = DIMENSION OF THE SQUARE Y TENSOR MATRIX
C     NUMPTS = NUMBER OF POINTS USED IN CALCULATING THE RHO MATRICES
C     NINDEX = TIME INDEX ARRAY USED IN DISSPLA PLOTS
C     PLTPTS = NUMBER OF DATA POINTS USED IN DISSPLA PLOTS
C     RANGE =  +/- RANGE OF UNIFORM WHITE NOISE
C     RHO = ARRAY OF REFLECTION COEFFICIENTS
C     RHOSUM = ARRAY USED IN CALCULATING THE AVERAGE RHO MATRIX
C     SCALE = RECIPROCAL OF THE NORM OF THE SYSTEM OUTPUT FOR WHEN
C             THE SYSTEM IS EXCITED BY WHITE NOISE
C     STDDEV = STANDARD DEVIATION OF GUASSIAN WHITE NOISE
C     TRIAL = NUMBER OF NOISE REALIZATIONS USED IN CALCULATING AVG RHO
C     X = INPUT INTO SYSTEM H(Z)
C     XHAT = LATTICE OUTPUT THAT APPROXIMATES X
C     XHPLOT = ARRAY OF REAL*4 VALUES OF XHAT USED IN DISSPLA PLOTS
C     XMAX,XMIN = RANGE OF X AND XHPLOT VALUES USED IN DISSPLA PLOTS
C     Y = OUTPUT OF H(Z)
C     YPLOT = ARRAY OF REAL*4 VALUES OF Y USED IN DISSPLA PLOTS
C     YMAX,YMIN = RANGE OF YPLOT VALUES USED IN DISSPLA PLOTS
C     YSUM = RUNNING TOTAL USED IN CALCULATING THE SYSTEM OUTPUT
C
C     ***VARIABLE DECLARATIONS****
      INTEGER TRIAL,NUMPTS,PLTPTS,IY,IM1,JM1
      REAL*4 X(5000),MSEMAX,RANGE,STDDEV,YMAX,YMIN,XMAX,XMIN,GRAFN
      REAL*4 XHPLOT(5000),NINDEX(5000),MSEPLT(5000),YPLOT(5000),GRAFP
      REAL*8 Y(5000),R(26,26),RHOSUM(26,26),XHAT(5000),NORM(26),MSESUM
      REAL*8 RHO(26,26),MSE(5000),DSEED,ALPHA(26,26),BETA(26,26),H(5,5)
      REAL*8 SCALE,YSUM
```

```
C
C     SET THE NUMBER OF WHITE NOISE REALIZATIONS USED TO CALCULATE
C           THE AVERAGE RHO MATRIX
      TRIAL = 25
C
C     DEFINE MODEL PARAMETERS
C
      N = 2
      MN = N * N
      MNP1 = MN + 1
C     DEFINE H(Z) COEFFICIENTS
C     THE AR PARAMETERS H(2,1)=0.6 AND H(1,2)=.08 CORRESPOND TO
C        GAMMA(2)=-.55 AND GAMMA(3)=-.08
      DO 7 I=1,N
         DO 6 J=1,N
            H(I,J) = 0.
6        CONTINUE
7     CONTINUE
C     H(1,1) =  0.5
      H(2,1) = 0.6
C     H(1,3) = 0.2
      H(1,2) = 0.08
C     H(2,2) = 0.2
C     H(2,3) = 0.5
C     H(3,1) = 0.2
C
C     INITIALIZE MSESUM, MSEMAX AND RHOSUM MATRIX
      DO 2 I=1,MNP1
         DO 1 J=1,MNP1
            RHOSUM(I,J) = 0.
1        CONTINUE
2     CONTINUE
      MSESUM = 0.
      MSEMAX = 0.
      XMAX = 0.
      YMAX = 0.
C     DEFINE SEED VALUES, SATURATION LIMIT, RANGE OF UNIFORM NOISE,
C     STD DEV OF GAUSS NOISE, AND NUMBER OF POINTS IN TIME SEQUENCES
      IY = 1354
      DSEED = 1243073.5D0
      SAT = 0.7
      RANGE = 1.73205-
      STDDEV = 1.0
      NUMPTS = 5000
      PLTPTS = 5000
      GRAFN = FLOAT(NUMPTS + 1)
      GRAFP = FLOAT(PLTPTS + 1)
C
C     WRITE HEADER FOR UNIFORM NOISE INPUT
      WRITE(8,4) RANGE
4     FORMAT(T1,'INPUT WHITE UNIFORM NOISE HAS ZERO MEAN AND RANGE OF
     *+/-  ',F6.3)
      WRITE(8,5) TRIAL,NUMPTS,IY
5     FORMAT(T1,'RHO IS AVERAGED OVER ',I3,' TRIALS OF ',I5,' POINTS.
     *  INITIAL SEED=',I6)
C
C     WRITE HEADER FOR GUASSIAN NOISE INPUT
C     WRITE(8,4) STDDEV
C4    FORMAT(T1,'INPUT WHITE GAUSS NOISE HAS ZERO MEAN AND STD DEV OF',
C    *F6.3)
C     WRITE(8,5) TRIAL,NUMPTS,DSEED
C5    FORMAT(T1,'RHO IS AVERAGED OVER ',I3,' TRIALS OF ',I5,' POINTS.
C    *  INITIAL SEED=',F10.1)
C
C     PRINT H MATRIX
      WRITE(8,8)
8     FORMAT(T1,'Y(K)=X(K)-(TENSOR PRODUCT H*Y) WHERE H(I,J) = ')
      DO 10 I=1,N
         WRITE(8,9) (H(I,J),J=1,N)
```

```
9         FORMAT(T10,5(F6.3,4X))
10      CONTINUE
C
C       ****************************************************************
C       RUN TRIALS FOR DIFFERENT SEED VALUES TO GET AVERAGE RHC MATRIX
C                    ******************************
        DO 50 L=1,TRIAL
           DSEED = DSEED/DFLOAT(L)
           IY = IY/L
C       SELECT EITHER A GUASSIAN OR UNIFORM INPUT NOISE SEQUENCE
        DO 11 K=1,NUMPTS
C         THE INPUT NOISE IS UNIFORM ON (-RANGE,RANGE) WITH
C         MEAN = 0 AND VARIANCE =((2*RANGE)**2)/12)
            X(K) = 2.0 * (URAND(IY) - .5) * RANGE
C         THE INPUT NOISE IS GAUSSIAN, MEAN=0 AND VARIANCE=STDDEV**2
C         X(K) = GGNQF(DSEED) * STDDEV
11      CONTINUE
C
        Y(1) = DBLE(X(1)) - H(1,1)
        Y(2) = DBLE(X(2)) - (H(1,1) + H(2,1)*Y(1) + H(3,1)*Y(1)*Y(1))
        DO 15 K=3,NUMPTS
           YSUM = 0.
           DO 14 I=1,N
                 DO 13 J=1,N
                       IM1 = I-1
                       JM1 = J-1
                       YSUM=H(I,J)*COORD(Y(K-1),IM1)*COORD(Y(K-2),JM1)+YSUM
13               CONTINUE
14         CONTINUE
           Y(K) = DBLE(X(K)) - YSUM
15      CONTINUE
C
C       ***CALL SUBROUTINES****
C       DETERMINE AUTOCORRELATION MATRIX FOR Y SEQUENCE
        CALL NLCLAT(Y,NUMPTS,R,N)
C       DETERMINE REFLECTION FACTORS FROM AUTOCORRELATION MATRIX
        CALL SCHUR(RHO,R,ALPHA,BETA,MNP1)
C       ADD TOGETHER THE RHO MATRICES FROM EACH TRIAL
        DO 45 I=1,MNP1
           DO 40 J=1,MNP1
                 RHOSUM(I,J) = RHO(I,J) + RHOSUM(I,J)
40         CONTINUE
45      CONTINUE
50      CONTINUE
C       CALCULATE AVERAGE RHO MATRIX AND TRUNCATE TO TWO DECIMALS
        DO 54 I=1,MNP1
           DO 53 J=1,MNP1
                 RHO(I,J) = RHOSUM(I,J)/DFLOAT(TRIAL)
                 RHO(I,J) = DINT(RHO(I,J)*100.)/100.
53         CONTINUE
54      CONTINUE
C
        WRITE(8,55)
55      FORMAT(T1,'RHO(I,J) =')
        DO 60 I = 1,MNP1
           WRITE(8,58)(RHO(I,J),J=1,MNP1)
58         FORMAT(T1,10F6.2)
60      CONTINUE
C
C
C       ***  NORMALIZE THE LATTICE INPUT SIGNALS AND PASS THE  ***
C            NOISE GENERATED DATA THROUGH THE LATTICE FILTER.
        CALL NORMS(Y,N,NUMPTS,NORM)
        SCALE = 1.0/NORM(1)
        CALL NLLAT(Y,RHO,N,NUMPTS,NORM,XHAT)
C
C       ***EXAMINE THE WHITENING EFFECT OF THE LATTICE FILTER BY  *****
C          CALCULATING THE MEAN-SQARE ERROR BETWEEN THE INPUT WHITE
C          NOISE AND THE FORWARD ERROR SIGNAL OUTPUT FROM THE TOP
```

```
C         ROW OF THE LATTICE.
          DO 63 K=1,NUMPTS
              MSESUM = (DBLE(X(K))-XHAT(K))*(DBLE(X(K))-XHAT(K)) + MSESUM
              MSE(K) =DSQRT(MSESUM/DFLOAT(K))
              MSEPLT(K) = SNGL(MSE(K))
              IF(MSEPLT(K).GT.MSEMAX) MSEMAX = MSEPLT(K)
63        CONTINUE
C
C         NOW THAT THE INVERSE FILTER HAS BEEN EVALUATED, DO THE
C         DECONVOLUTION PROBLEM.  FIRST, DEFINE THE "UNKNOWN" INPUT
C         SEQUENCE X(K) AND THEN GENERATE THE SYSTEM OUTPUT Y(K).
C
          DO 85 K=1,PLTPTS
              IF(K.LE.250) X(K) = FLOAT(K)/250.
              IF(K.LE.750.AND.K.GT.250) X(K) = 2.0 - FLOAT(K)/250.
              IF(K.LE.1250.AND.K.GT.750) X(K) = FLOAT(K)/250. - 4.0
              IF(K.LE.1750.AND.K.GT.1250) X(K) = 1.0 - FLOAT(K-1250)/250.
              IF(K.LE.2000.AND.K.GT.1750) X(K) = FLOAT(K-1750)/250. - 1.0
              IF(K.LE.2400.AND.K.GT.2000) X(K) = 0.5
              IF(K.LE.2700.AND.K.GT.2400) X(K) = -0.5
              IF(K.LE.2900.AND.K.GT.2700) X(K) = 0.5
              IF(K.LE.3000.AND.K.GT.2900) X(K) = -0.5
              IF(K.LE.3500.AND.K.GT.3000)X(K)= 0.7*SIN(0.0126*FLOAT(K-3001))
              IF(K.LE.5000.AND.K.GT.3500)X(K)= 0.9*SIN(0.0021*FLOAT(K-3501))
85        CONTINUE
C
          Y(1) = DBLE(X(1)) - H(1,1)
          Y(2) = DBLE(X(2)) - (H(1,1) + H(2,1)*Y(1) + H(3,1)*Y(1)*Y(1))
          DO 90 K=3,PLTPTS
              YSUM = 0.
              DO 89 I=1,N
                    DO 88 J=1,N
                          IM1 = I-1
                          JM1 = J-1
                          YSUM=H(I,J)*COORD(Y(K-1),IM1)*COORD(Y(K-2),JM1)+YSUM
88                  CONTINUE
89            CONTINUE
              Y(K) = DBLE(X(K)) - YSUM
90        CONTINUE
C
CC        NOW PASS THE SYSTEM OUTPUT DATA THROUGH THE LATTICE FILTER
CC        EMBEDDED WITH THE PREVIOUSLY CALCULATED REFLECTION FACTORS
CC        RHO(I,J).  THE FORWARD ERROR SIGNAL OUT OF THE TOP ROW OF THE
CC        LATTICE IS XHAT(K), AND SHOULD APPROXIMATE THE DESIRED SIGNAL
C         X(K) AFTER IT IS RENORMALIZED AND SCALED.
          CALL NORMS(Y,N,PLTPTS,NORM)
          CALL NLLAT(Y,RHO,N,PLTPTS,NORM,XHAT)
          DO 98 K=1,PLTPTS
              XHAT(K) = SCALE * (XHAT(K)*NORM(1))
              XHPLOT(K) = SNGL(XHAT(K))
              YPLOT(K) = SNGL(Y(K))
              IF(ABS(X(K)).GT.XMAX) XMAX = ABS(X(K))
              IF(ABS(XHPLOT(K)).GT.XMAX) XMAX = ABS(XHPLOT(K))
              IF(ABS(YPLOT(K)).GT.YMAX) YMAX = ABS(YPLOT(K))
98        CONTINUE
          XMIN = -1.0 * XMAX
          YMIN = -1.0 * YMAX
          DO 99 K=1,5000
              NINDEX(K) = FLOAT(K-1)
99        CONTINUE
C
CC        **** DISSPLA PLOTTING ROUTINES ****
C
CC        PLOT SYSTEM INPUT AND LATTICE OUTPUT
C         CALL TEK618
          CALL COMPRS
          CALL RESET('ALL')
          CALL PAGE(8.0,6.5)
```

```
        CALL XINTAX
        CALL AREA2D(6.75,5.0)
        CALL XNAME('TIME SAMPLES$',100)
        CALL YNAME('X(K),XHAT(K)$',100)
C       CALL YNAME('X(K)$',100)
        CALL CROSS
        CALL GRAF(0.,500.,GRAFP,XMIN,'SCALE',XMAX)
        CALL LINESP (2.0)
        CALL LINES('X(K)$',IPAK,1)
        CALL LINES('XHAT(K)$',IPAK,2)
        CALL LEGLIN
        CALL DOT
        CALL CURVE(NINDEX,X,PLTPTS,0)
        CALL RESET('DOT')
        CALL CURVE(NINDEX,XHPLOT,PLTPTS,0)
        CALL LEGEND (IPAK,2,5.0,0.5)
        CALL ENDPL(0)
C
C       PLOT Y
        CALL AREA2D(6.75,5.0)
        CALL XNAME('TIME SAMPLES$',100)
        CALL YNAME('Y(K)$',100)
C       CALL HEADIN('Y(K) = X(K) - 0.2*Y(K-1)**2$',100,1.5,1)
        CALL HEADIN('Y(K) = X(K)-(0.6*Y(K-1)+0.08*Y(K-2))$',100,1.5,1)
C       CALL HEADIN('Y(K)=X(K)-(0.1*Y(K-1)+ 0.5*Y(K-1)*Y(K-2)**2)$',
C      *100,1.5,1)
        CALL CROSS
        CALL GRAF(0.,500.,GRAFP,YMIN,'SCALE',YMAX)
        CALL CURVE(NINDEX,YPLOT,PLTPTS,0)
        CALL ENDPL(0)
C
C       PLOT MSE
        CALL AREA2D(6.75,5.0)
        CALL XNAME('TIME SAMPLES$',100)
        CALL YNAME('MSE(K)$',100)
        CALL GRAF(0.,500.,GRAFN,0.,'SCALE',MSEMAX)
        CALL CURVE(NINDEX,MSEPLT,NUMPTS,0)
        CALL ENDPL(0)
        CALL DONEPL
C
        STOP
        END
C*************************************************************************
C
C       SUBROUTINE SCHUR
C
C       CALCULATES THE REFLECTION FACTORS FROM THE CORRELATION MATRIX
C
C       WRITTEN 29 APRIL 1985 BY P.J. LENK (REF 12:P. 174)
C
C*************************************************************************
        SUBROUTINE SCHUR(RHO,R,ALPHA,BETA,N)
        REAL*8 RHO(26,26),R(26,26),ALPHA(26,26),BETA(26,26),RNORM,T
C
C       INITIALIZE THE ALPHA AND BETA ARRAYS
C
        DO 10 I = 1,N
            DO 5 J = 1,N
                ALPHA(I,J) = R(I,J)/DSQRT(R(I,I))
                BETA(I,J) = ALPHA(I,J)
                RHO(I,J) = 0.0
5         CONTINUE
C          WRITE(8,7)(ALPHA(I,J),J=1,N)
C7         FORMAT(5(2X,E12.5))
10      CONTINUE
C
C       BEGIN CALCULATING THE REFLECTION FACTORS
C
        DO 50 J = 2,N
```

```
            NJ1 = N - J + 1
            DO 40 I = 1,NJ1
                JI1 = J + I - 1
                IP1 = I + 1
                RHO(I,JI1) = ALPHA(I,JI1)/BETA(IP1,JI1)
                RNORM = DSQRT(1.0 - RHO(I,JI1)*RHO(I,JI1))
                DO 30 K = 1,N
                    T = ALPHA(I,K)
                    ALPHA(I,K) = (ALPHA(I,K)-RHO(I,JI1)*BETA(IP1,K))/RNORM
                    BETA(I,K) = (BETA(IP1,K)-RHO(I,JI1)*T)/RNORM
30          CONTINUE
40      CONTINUE
C        WRITE(8,42)J,((ALPHA(I,K),K=1,N),I=1,N)
C42      FORMAT(/2X,I3,4(2X,E12.5))
C        WRITE(8,42)J,((BETA(I,K),K=1,N),I=1,N)
50    CONTINUE
C
C
        RETURN
        END
C*******************************************************************************
C
C     FUNCTION URAND
C
C     TAKEN FROM "COMPUTER METHODS FOR MATHEMATICAL COMPUTATIONS" BY
C     G.E. FORSYTHE, M.A. MALCOLM, AND C.B. MOLER
C
C*******************************************************************************
C
        REAL FUNCTION URAND(IY)
        INTEGER IA,IC,ITWO,M2,M,MIC
        DOUBLE PRECISION HALFM
        REAL S
        DOUBLE PRECISION DATAN,DSQRT
        DATA M2/0/,ITWO/2/
        IF(M2.NE.0) GO TO 20
C
C     IF FIRST ENTRY, COMPUTE MACHINE INTEGER WORD LENGTH
C
        M=1
    10  M2=M
        M=ITWO*M2
        IF(M.GT.M2) GO TO 10
        HALFM=M2
C
C     COMPUT MULTIPLIER AND INCREMENT FOR LINEAR CONGRUENTIAL METHOD
C
        IA=8*IDINT(HALFM*DATAN(1.D0)/8.D0)+5
        IC=2*IDINT(HALFM*(.5D0-DSQRT(3.D0)/6.D0))+1
        MIC=(M2-IC)+M2
C
C     S IS THE SCALE FACTOR FOR CONVERTING TO FLOATING POINT
C
        S=.5/HALFM
C
C     COMPUTE NEXT RANDOM NUMBER
C
    20  IY=IY*IA
C
C     THE FOLLOWING STATEMENT IS FOR COMPUTERS WHICH DO NOT ALLOW
C     INTEGER OVERFLOW ON ADDITION
C
        IF(IY.GT.MIC) IY=(IY-M2)-M2
C
        IY=IY+IC
C
C     THE FOLLOWING IS FOR COMPUTERS FOR WHICH THE WORD LENGTH
C     FOR ADDITION IS GREATER THAN FOR MULTIPLICATION
C
```

```
      IF(IY/2.GT.M2)IY=(IY-M2)-M2
C
CC    THE FOLLOWING STATEMENT IS FOR COMPUTERS WHERE INTEGER OVERFLOW
CC    AFFECTS THE SIGN BIT
C
      IF(IY.LT.0)IY=(IY+M2)+M2
      URAND=FLOAT(IY)*S
      RETURN
      END
C*****************************************************************************
C                                                                             *
CC      SUBROUTINE NLCLAT                                                     *
CC                                                                            *
CC      THIS SUBROUTINE PRODUCES A CORRELATION MATRIX FROM NONLINEAR          *
CC      TIME SEQUENCE IN AN ORDER WHICH IS COMPATIBLE WITH SUBROUTINE         *
CC      SCHUR.                                                                *
CC                                                                            *
CC      WRITTEN 7 MAY 1985 BY P.J. LENK (REF 12:P. 181)                       *
C                                                                             *
C*****************************************************************************
      SUBROUTINE NLCLAT (Y,IYS,R,N)
      REAL*8 Y(5000),R(26,26),SUM,VEC(26)
C
CC    DEFINE CONSTANTS
C
      MN = N*N
      MNP1 = MN + 1
      IYSM2 = IYS - 2
      FIYSM2 = FLOAT(IYSM2)
C
CC    INITIALIZE R MATRIX TO ZERO
C
      DO 20 I = 1,MNP1
         DO 10 J = 1,MNP1
            R(I,J) = 0.0
10       CONTINUE
20    CONTINUE
C
CC    BEGIN OUTER LOOP
C
      DO 80 I = 3,IYS
         IR = 1
         VEC(IR) = Y(I)
         DO 50 MP1 = 1,N
            MO = MP1 - 1
            LLIM = 2*MP1 - 1
            DO 40 L = 1,LLIM
               LO = L - 1
               I1 = MO
               J1 = LO/2
               IF (MOD(LO,2).EQ.0) GO TO 30
               I1 = J1
               J1 = MO
30             IR = IR + 1
               VEC(IR) = COORD(Y(I-1),I1)*COORD(Y(I-2),J1)
40          CONTINUE
50       CONTINUE
C
CC    CALCULATE THE CORRELATIONS
C
         DO 70 J = 1,MNP1
            DO 60 K = J,MNP1
               R(J,K) = R(J,K) + VEC(J)*VEC(K)
C              WRITE(6,12)VEC(J),VEC(K),R(J,K)
C12            FORMAT(3(2X,E12.5))
60          CONTINUE
70       CONTINUE
80    CONTINUE
C
```

```
C      DIVIDE BY THE NUMBER OF DATA ELEMENTS CONSIDERED
C
       DO 100 J = 1,MNP1
          DO 90 K = J,MNP1
             R(J,K) = R(J,K)/FIYSM2
90        CONTINUE
100    CONTINUE
C
CC     FILL IN THE SYMMETRIC HALF OF CORRELATION MATRIX
C
       DO 120 I = 2,MNP1
          IM1 = I - 1
          DO 110 J = 1,IM1
             R(I,J) = R(J,I)
110       CONTINUE
120    CONTINUE
C
CC
       RETURN
       END
C
CC*****************************************************************************
CCCC
CCCC   FUNCTION COORD
CCCC
CCCC   GENERATES OUTPUT OF RANDOM FUNCTION
CCCC   CREATED 23 AUG 84  (REF 12:P. 187)
CC*****************************************************************************
       DOUBLE PRECISION FUNCTION COORD(X,I)
C      USE SIMPLE POWER SERIES TYPE POLYNOMIALS
C
       Y = 1.0
       IF (I.EQ.0) GO TO 30
       Y = X**I
30     COORD = DBLE(Y)
       RETURN
       END
C
CC*****************************************************************************
CCCC
CCCC   SUBROUTINE NLLAT
CCCC
CCCC   THIS SUBROUTINE IMPLEMENTS THE NONLINEAR LATTICE FILTER USING
CCCC   PREVIOUSLY CALCULATED REFLECTION FACTORS.
CCCC
CCCC   WRITTEN 30 APRIL 86
CC*****************************************************************************
       SUBROUTINE NLLAT(Y,RHO,N,NUMPTS,NORM,XHAT)
C      ***VARIABLE DEFINITIONS****
CCCC
CCCC   INFWD(I) = FORWARD ERROR INPUT INTO THE I'TH ROW OF THE LATTICE
CCCC   INBKD(I) = BACKWARD   "     "     "   "   "   "   "   "   "
CCCC   OUTFWD(I)= FORWARD ERROR OUTPUT FROM THE I'TH ROW OF THE LATTICE
CCCC   OUTBKD(I)= BACKWARD   "     "     "   "   "   "   "   "   "
CCCC   Y = INPUT DATA VECTOR
CCCC   RHO = REFLECTION FACTOR MATRIX
CCCC   NORM = VECTOR OF NORMS OF THE LATTICE INPUT TERMS
CCCC   XHAT = OUTPUT DATA VECTOR; IT IS THE FORWARD ERROR SIGNAL FROM
CCCC          THE LAST STAGE OF THE FIRST ROW
CCCC   NUMPTS = NUMBER OF POINTS IN THE INPUT/OUTPUT SEQUENCES
CCCC   N = DIMENSION OF SQUARE Y DATA MATRIX
CCCC   MNP1 = DIMENSION OF THE RHO, NORM, AND INPUT/OUTPUT ARRAYS
CCC
C      ***VARIABLE DECLARATIONS****
       INTEGER N,MN,NUMPTS,LAST,MNP1
```

```
      REAL*8 Y(5000),RHO(26,26),NORM(26),XHAT(5000),INFWD(26)
      REAL*8 INBKD(26),OUTFWD(26),OUTBKD(26),RNORM
C
      MN = N * N
      MNP1 = MN + 1
C     INITIALIZE THE INPUT AND OUTPUT ARRAYS TO ZERO
      DO 5 I=1,26
         INFWD(I) = 0.
         INBKD(I) = 0.
         OUTFWD(I) = 0.
         OUTBKD(I) = 0.
5     CONTINUE
C
C     DETERMINE THE LATTICE INPUTS FOR EACH TIME K.
      DO 60 K=1,NUMPTS
C       CLEAR INPUTS FOR THIS TIME ITERATION
        DO 10 I=1,MNP1
              INFWD(I) = 0.
              INBKD(I) = 0.
10      CONTINUE
        IF(K.EQ.1) GO TO 15
        IF(K.EQ.2) GO TO 20
C       SET LATTICE INPUTS FOR CASE WHEN K>=3
        IR = 1
        INFWD(IR) = Y(K)/NORM(IR)
        DO 13 MP1 = 1,N
              MO = MP1 - 1
              LLIM = 2*MP1 - 1
              DO 12 L = 1,LLIM
                    LO = L - 1
                    I1 = MO
                    J1 = LO/2
                    IF (MOD(LO,2).EQ.0) GO TO 11
                    I1 = J1
                    J1 = MO
11                IR = IR + 1
                    INFWD(IR)=COORD(Y(K-1),I1)*COORD(Y(K-2),J1)/NORM(IR)
12            CONTINUE
13      CONTINUE
C       INFWD(1) = Y(K)/NORM(1)
C       INFWD(2) = 1./NORM(2)
C       INFWD(3) = Y(K-1)/NORM(3)
C       INFWD(4) = Y(K-2)/NORM(4)
C       INFWD(5) = Y(K-1)*Y(K-2)/NORM(5)
C       INFWD(6) = Y(K-1)*Y(K-1)/NORM(6)
C       INFWD(7) = Y(K-2)*Y(K-2)/NORM(7)
C       INFWD(8) = Y(K-1)*Y(K-1)*Y(K-2)/NORM(8)
C       INFWD(9) = Y(K-1)*Y(K-2)*Y(K-2)/NORM(9)
C       INFWD(10) = Y(K-1)*Y(K-1)*Y(K-2)*Y(K-2)/NORM(10)
        DO 14 J=1,MN
              INBKD(J) = INFWD(J+1)
14      CONTINUE
        GO TO 25
C
C       SET INPUTS FOR K=1 CASE
15      INFWD(1) = Y(1)/NORM(1)
        INFWD(2) = 1./NORM(2)
        INBKD(1) = INFWD(2)
        GO TO 25
C
C       SET INPUTS FOR K=2 CASE
20      INFWD(1) = Y(2)/NORM(1)
        INFWD(2) = 1./NORM(2)
        INFWD(3) = Y(1)/NORM(3)
        INFWD(6) = Y(1)*Y(1)/NORM(6)
        INFWD(11) = Y(1)*Y(1)*Y(1)/NORM(11)
        INFWD(18) = Y(1)*Y(1)*Y(1)*Y(1)/NORM(18)
        INBKD(1) = INFWD(2)
        INBKD(2) = INFWD(3)
```

```
            INBKD(5) = INFWD(6)
            INBKD(10) = INFWD(11)
            INBKD(17) = INFWD(18)
C
C       IMPLEMENT THE J'TH UPPER DIAGONAL OF THE LATTICE. MOVE ALONG THE
C       DIAGONAL BY STARTING AT ROW L=1 AND MOVING DOWN TO ROW L=LAST.
25          DO 50 J=1,MN
                LAST = MNP1 - J
                IF(J.EQ.1) GO TO 40
C           IF NOT ON THE FIRST UPPER DIAGONAL (I.E. J NOT = 1) THEN UPDATE
C           THE INPUTS FROM THE OUTPUTS OF THE PREVIOUS DIAGONAL
                DO 35 L=1,LAST
                    INFWD(L) = OUTFWD(L)
                    INBKD(L) = OUTBKD(L+1)
35              CONTINUE
C               DO THE LATTICE CALCULATIONS
40              DO 45 L=1,LAST
                    RNORM = DSQRT(1.0 - RHO(L,J+L)*RHO(L,J+L))
                    OUTFWD(L) = (INFWD(L) - RHO(L,J+L)*INBKD(L))/RNORM
                    OUTBKD(L) = (INBKD(L) - RHO(L,J+L)*INFWD(L))/RNORM
45              CONTINUE
50          CONTINUE
C           REMOVE NORMALIZATION FACTOR FROM THE OUTPUT OF THE TOP ROW
              XHAT(K) = OUTFWD(1)
60      CONTINUE
        RETURN
        END
C
CCCCC
CCCCC*****************************************************************************
CCCCC
CCCCC       SUBROUTINE NORMS
CCCCC
CCCCC       THIS SUBROUTINE CALCULATES THE NORMS OF THE INPUTS
CCCCC       TO THE NONLINEAR LATTICE.
CCCCC
CCCCC       WRITTEN 30 APRIL 86
CCCCC*****************************************************************************
        SUBROUTINE NORMS(Y,N,NUMPTS,NORM)
C
CC
        INTEGER NUMPTS,N,MN,MNP1,I,K,IR,LLIM,L0,M0,MP1,I1,J1,J,L
        REAL*8 VEC(26),NORM(26),Y(5000)
        MN = N * N
        MNP1 = MN + 1
C       INITIALIZE VECTORS TO ZERO
        DO 10 K = 1,MNP1
            VEC(K) = 0.
            NORM(K) = 0.
10      CONTINUE
C
CC      SINCE TIME INDEX STARTS AT I=3, INITIALIZE NORMS OF INPUTS WHICH
C       ARE POWERS OF Y(I-1) TO ACCOUNT FOR TIMES I=1 AND I=2.
        NORM(1) = Y(1) * Y(1) + Y(2) * Y(2)
        NORM(2) = 1.0 +1.0
        NORM(3) = Y(1) * Y(1)
        NORM(6) = NORM(3) * NORM(3)
        NORM(1) = NORM(6) * NORM(3)
        NORM(18) = NORM(6) * NORM(6)
C
CC      FOR TIME I, FORM A Y DATA VECTOR WHOSE COMPONENTS MATCH THE
C       INPUTS TO THE NONLINEAR LATTICE.
        DO 80 I = 3,NUMPTS
            IR = 1
            VEC(IR) = Y(I)
```

```
            DO 50 MP1 = 1,N
                  MO = MP1 - 1
                  LLIM = 2*MP1 - 1
                  DO 40 L = 1,LLIM
                        LO = L - 1
                        I1 = MO
                        J1 = LO/2
                        IF (MOD(LO,2).EQ.0) GO TO 30
                        I1 = J1
                        J1 = MO
30                   IR = IR + 1
                        VEC(IR) = COORD(Y(I-1),I1)*COORD(Y(I-2),J1)
40                CONTINUE
50          CONTINUE
C
C           CALCULATE THE NORMS
              DO 60 K=1,MNP1
C                 IF(K.EQ.2) GO TO 60
                  NORM(K) = NORM(K) + VEC(K)*VEC(K)
60          CONTINUE
80       CONTINUE
C
            WRITE(8,83)
83       FORMAT(T3,'K',T10,'NORM(K)')
            DO 85 K=1,MNP1
               NORM(K) = DSQRT(NORM(K)/DFLOAT(NUMPTS))
               WRITE(8,86) K,NORM(K)
86          FORMAT(T1,I3,T8,E12.5)
85       CONTINUE
            RETURN
            END
```

13. Ljung, L. and Soderstrom, T., Theory and Practice of Recursive Identification, The M.I.T. Press, 1983.

14. Schetzen, M., The Volterra and Wiener Theories of Nonlinear Systems, John Wiley & Sons, Inc., 1980.

BIBLIOGRAPHY

Papoulis,A., Probability, Random Variables, and Stochastic Processes, McGraw-Hill, Inc., 1984.

Oppenheim, A. and Schafer R.W.,Digital Signal Processing, Prentice-Hall, Inc., 1975.

Parker, S.R., An Integrated Approach to Discrete Processing for Scientists and Engineers unpublished lecture notes presented at the Naval Postgraduate School, Monterey, California, 1983.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Department Chairman, Code 62 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943	1
4.	Dr. S.R. Parker, Code 62Px Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943	10
5.	Dr. L.J. Ziomek, Code 62Zm Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943	1
6.	Lt(N) P.J. Lenk DMCS 3-2-8 DGMEM. CEM, NDHQ 101 Colonel By Drive, Ottawa, Ontario Canada, K1V 0K2	1
7.	MAJ Edward M. Siomacco 1285 Leahy Road Monterey, California 93940	1
8.	LT Scot L. Johnson 3806 Urban Ave. N. Brooklyn Center, Minnesota 55429	2

END

2-87

DTIC